

Requested Patent: JP2202637A

Title:

MEMORY MANAGEMENT IN HIGH-PERFORMANCE FAULT-TOLERANT  
COMPUTER SYSTEM. ;

Abstracted Patent: EP0372578, A3 ;

Publication Date: 1990-06-13 ;

Inventor(s):

PEET CHARLES E JR; HORST ROBERT W; MEHTA NIKHIL A; ALLISON JOHN  
DAVID; CUTTS RICHARD W JR; DEBACKER KENNETH C; JEWETT DOUGLAS  
E ;

Applicant(s): TANDEM COMPUTERS INC (US) ;

Application Number: EP19890122707 19891208 ;

Priority Number(s):

US19880282469 19881209; US19880282540 19881209; US19880283573  
19881213 ;

IPC Classification: G06F11/16; G06F11/18 ;

Equivalents: AU5202790, AU628497, CA2003342, JP7013789

ABSTRACT:

A computer system in a fault-tolerant configuration employs three identical CPUs executing the same instruction stream, with two identical, self-checking memory modules storing duplicates of the same data. Memory references by the three CPUs are made by three separate busses connected to three separate ports of each of the two memory modules. The three CPUs are loosely synchronized, as by detecting events such as memory references and stalling any CPU ahead of others until all execute the function simultaneously; interrupts can be synchronized by ensuring that all three CPUs implement the interrupt at the same point in their instruction stream. Memory references via the separate CPU-to-memory busses are voted at the three separate ports of each of the memory modules. I/O functions are implemented using two identical I/O busses, each of which is separately coupled to only one of the memory modules. A number of I/O processors are coupled to both I/O busses. Each CPU has its own fast cache and also local memory not accessible by the other CPUs. A hierarchical virtual memory management arrangement for this system employs demand paging to keep the most-used data in the local memory, page-swapping with the global memory. Page swapping with disk memory is through the global memory; the global memory is used as a disk buffer and also to hold pages likely to be needed for loading to local memory. The operating system kernel is kept in local memory. A private-write area is included in the shared memory space in the memory modules to allow functions such as software voting of state information unique to CPUs. All CPUs write state information to their private-write area, then all CPUs read all the private-write areas for functions such as detecting differences in interrupt cause or the like.

## ⑫ 公開特許公報(A) 平2-202637

⑤ Int. Cl.<sup>5</sup>G 06 F 11/18  
15/16

識別記号

3 1 0 E  
3 3 0 Z  
4 7 0 J

庁内整理番号

7368-5B  
6745-5B  
6745-5B

④ 公開 平成2年(1990)8月10日

審査請求 未請求 請求項の数 44 (全47頁)

⑥ 発明の名称 フォールトトレラントコンピュータにおけるメモリ管理システム

⑦ 特 願 平1-322462

⑧ 出 願 平1(1989)12月11日

優先権主張 ⑨ 1988年12月9日 ⑩ 米国(US) ⑪ 282,469

⑫ 発 明 者 リチャード・ダブリュ アメリカ合衆国 78626 テキサス、ジョージタウン、エ  
ー・カツツ・ジュニア ルム・ストリート 1312番⑬ 発 明 者 チャールズ・イー・ピ アメリカ合衆国 78753 テキサス、オースチン、オーク  
ート・ジュニア ブルック・ドライブ 11920番⑭ 出 願 人 タンデム・コンピュ アメリカ合衆国 95014 カリフォルニア、キューバーテ  
ターズ・インコーポレ イノ、ノース・タンタウ・アベニュー 10435番  
イテッド⑮ 代 理 人 弁理士 青山 蓑 外1名  
最終頁に続く

## 明 細 書

## 1. 発明の名称

フォールトトレラントコンピュータにおけるメ  
モリ管理システム

## 2. 特許請求の範囲

(1) それぞれ同一の命令ストリームを実行し、  
ページングとともに仮想メモリのアドレス指定を  
用いる多重CPUを備え、上記各CPUは、該CPUによってのみアクセ  
ス可能であり選択されたページを収容するローカ  
ルメモリを有し、すべての上記CPUによってアクセス可能なグ  
ローバルメモリを備え、上記ローカルメモリは上記グローバルメモリよ  
りも短いアクセス時間を有し、上記グローバルメ  
モリは選択されたページを収容し、最も使用され  
るページを上記各CPUの上記ローカルメモリに  
おいて維持するための要求時に上記ローカルメモ  
リとの間でページのスワッピングが行われること  
を特徴とするコンピュータシステム。(2) 上記システムはさらに、上記グローバルメ  
モリに接続されかつ上記グローバルメモリよりも  
長いアクセス時間を有するディスクメモリを備え、  
上記ディスクメモリは、上記仮想メモリのアドレ  
ス指定によって定義されたページを収容し、要求  
時に上記グローバルメモリとローカルメモリとの  
間でページのスワッピングが行われることを特徴  
とする請求項1記載のシステム。(3) 上記システムはさらに、上記各CPUのた  
めに上記ローカルメモリに格納された核を有する  
オペレーティングシステムを備えたことを特徴と  
する請求項1記載のシステム。(4) 上記各CPUは、上記ローカルメモリのア  
クセス時間よりも短いアクセス時間を有する独立  
したキャッシュメモリを有することを特徴とする  
請求項1記載のシステム。(5) 上記複数のCPUに互いに独立にクロック  
が供給され、上記複数のCPUは上記グローバル  
メモリをアクセスする時に同期化され、上記グロ  
ーバルメモリは2重化されることを特徴とする請

求項1記載のシステム。

(6) 上記グローバルメモリは、上記グローバルメモリを介してのみアクセス可能である入出力手段に接続され、上記グローバルメモリは上記複数のCPUによる入出力要求をステージングするために用いられることを特徴とする請求項1記載のシステム。

(7) ページングとともに仮想メモリのアドレス指定を用いて同一の命令ストリームを多重CPUにおいて実行するステップと、

上記命令ストリームの実行において上記各CPUによってローカルメモリをアクセスするステップを備え、上記各ローカルメモリは選択されたページを上記ローカルメモリに格納するために上記複数のCPUの1つによってのみアクセス可能であり、

上記命令ストリームの実行時にすべての上記CPUによってグローバルメモリをアクセスするステップを備え、

上記グローバルメモリはすべてのCPUによ

含むことを特徴とする請求項7記載の方法。

(10) 上記各CPUは、上記ローカルメモリのアクセス時間よりも短いアクセス時間を有する独立したキャッシュメモリを有することを特徴とする請求項7記載の方法。

(11) 上記複数のCPUに互いに独立にクロックを供給するステップを含み、上記グローバルメモリをアクセスする時に上記複数のCPUを同期化するステップを含み、上記グローバルメモリは2重化されることを特徴とする請求項7記載の方法。

(12) 上記グローバルメモリは、上記グローバルメモリを介してのみアクセス可能な入出力手段に接続され、ステージングのために上記グローバルメモリを用いて上記複数のCPUと上記入出力手段との間でデータの転送を行うステップを含むことを特徴とする請求項7記載の方法。

(13) 核を有するオペレーティングシステムの制御のもとでページングとともに仮想メモリのアドレス指定を用いてマルチプロセッサにおいて同

てアクセス可能であって、要求時に上記ローカルメモリとの間でページのスワッピングが行われるグローバルメモリに選択されたページを格納し上記各CPUの上記ローカルメモリにおいて最も使用されるページを維持するために、上記ローカルメモリはグローバルメモリよりも短いアクセス時間を有することを特徴とするコンピュータシステムを動作させるための方法。

(8) 上記方法はさらに、上記グローバルメモリに接続されるディスクメモリにページを格納するステップを含み、上記ディスクメモリは上記グローバルメモリよりも長いアクセス時間を有し、上記ディスクメモリに格納されるページは、上記仮想メモリのアドレス指定によって定義されかつ要求時に上記グローバルメモリとローカルメモリとの間でページのスワッピングが行われることを特徴とする請求項7記載の方法。

(9) 上記各CPUのための上記ローカルメモリに格納される核を有するオペレーティングシステムのもとで上記命令ストリームを実行することを

一の命令ストリームを実行するステップと、上記命令ストリームの実行時に各プロセッサによってローカルメモリをアクセスするステップを備え、上記各ローカルメモリは、上記ローカルメモリに選択されたページを格納しかつ上記オペレーティングシステムの上記核を格納するために上記複数のプロセッサの1つによってのみアクセス可能であり、

上記命令ストリームの実行時にすべての上記プロセッサによって2重化されたグローバルメモリをアクセスするステップを備え、上記グローバルメモリはすべての上記プロセッサによってアクセス可能であり、上記ローカルメモリは、最も使用されるページを上記各プロセッサの上記ローカルメモリにおいて維持するために上記オペレーティングシステムの制御のもとで要求時に上記ローカルメモリとの間でページのスワッピングが行われた上記グローバルメモリに選択されたページを格納するために、上記グローバルメモリよりも短いアクセス時間を有し、

上記グローバルメモリに接続されるディスクメモリにページを格納するステップを備え、上記ディスクメモリは上記グローバルメモリよりも長いアクセス時間を有し、上記ディスクメモリに納されるページは、上記オペレーティングシステムを用いる上記仮想メモリのアドレス指定によって定義され、かつ要求時に上記グローバルメモリと上記ローカルメモリとの間でページのスワッピングが行われることを特徴とするコンピュータシステムを動作させるための方法。

(14) 上記各プロセッサは、上記ローカルメモリのアクセス時間よりも短いアクセス時間を有する独立したキャッシュメモリを有することを特徴とする請求項13記載の方法。

(15) 上記複数のプロセッサに互いに独立にクロックを供給するステップを含み、また、上記グローバルメモリに対するアクセス時に上記複数のプロセッサを同期化するステップを含むことを特徴とする請求項13記載の方法。

(16) 上記グローバルメモリは上記グローバル

シュコントローラを備えたことを特徴とするコンピュータシステム。

(18) 上記リフレッシュコントローラは、上記各CPUにおける上記命令ストリームにおいて同一の命令を実行する時に上記リフレッシュを初期化することを特徴とする請求項17記載のシステム。

(19) 上記複数のCPUは、すべての上記CPUによってアクセス可能な共通のメモリに対するアクセスをポートすることによって緩く同期化されることを特徴とする請求項17記載のシステム。

(20) 3個の上記CPUが存在し、上記3個のCPUは2重化された共通のグローバルメモリにアクセスすることを特徴とする請求項17記載のシステム。

(21) 命令ストリームを実行するCPUを備え、上記CPUは実行サイクルを備えるためにクロックが供給され、上記CPUはいくつかの命令実行の履行を待機する間にストールサイクルを実行し、

上記CPUは、実行サイクルを計数するがス

メモリを介してのみアクセス可能である入出力手段に接続され、ステージングのために上記グローバルメモリを用いて上記複数のプロセッサと上記入出力手段との間でデータを転送するステップを含むことを特徴とする請求項13記載の方法。

(17) それぞれ命令ストリームを実行する複数のCPUを備え、上記複数のCPUは複数の実行サイクルを備えるために互いに独立にクロックが供給され、上記複数のCPUはいくつかの命令実行の履行を待機する間ストールサイクルを実行し、

上記各CPUは、実行サイクルを計数するがストールサイクルを計数しない第1のカウンタを有するとともに、ストールサイクルを計数するための第2のカウンタを有し、

上記各CPUは周期的なリフレッシュを要求するローカルメモリを有し、

上記第1と第2のカウンタにตอบสนองし上記ローカルメモリのリフレッシュを初期化し上記第2のカウンタの出力に依存して所定数のリフレッシュサイクルを実行するための各CPUのためのリフレ

ールサイクルを計数しない第1のカウンタを有するとともに、ストールサイクルを計数するための第2のカウンタを有し、

上記CPUは周期的なリフレッシュを要求するメモリを有し、

上記第1と第2のカウンタにตอบสนองして上記メモリのリフレッシュを初期化し上記第2のカウンタの出力に依存して所定数のリフレッシュサイクルを実行するための上記CPUのためのリフレッシュコントローラを備えたことを特徴とするコンピュータシステム。

(22) 第3のカウンタは、上記第2のカウンタがオーバーフローする回数を計数し、上記リフレッシュサイクルの数は上記第3のカウンタの内容によって決定されることを特徴とする請求項21記載のシステム。

(23) 上記第1のカウンタは、与えられた時間周期において上記ローカルメモリによって必要なリフレッシュサイクルの数に關係するサイズを有することを特徴とする請求項21記載のシステム。

(24) 複数のCPUのそれぞれにおいて命令ストリームを実行するステップを備え、上記複数のCPUは複数の実行サイクルを備えるために互いに独立にクロックが供給され、上記複数のCPUはいくつかの命令実行の履行を待機する間にストールサイクルを実行し、

第1のカウントにおいて上記各CPUにおける実行サイクルを計数するがストールサイクルを計数せず、第2のカウントにおいて上記各CPUにおけるストールサイクルを計数するステップと、

上記各CPUが周期的なリフレッシュを要求するローカルメモリをアクセスするステップと、

上記第1と第2のカウントにตอบสนองして上記第2のカウントの出力に依存して所定数のリフレッシュサイクルを実行するために上記各CPUのための上記ローカルメモリのリフレッシュを初期化するステップとを備えたことを特徴とするコンピュータシステムを動作させるための方法。

(25) 上記リフレッシュを初期化するステップは、上記各CPUにおける上記命令ストリームに

によって上記各CPUによって上記状態情報に対して同じであるか否かの評価を行うことを特徴とするコンピュータシステム。

(29) 複数の上記プライベートメモリ空間が存在し、上記プライベートメモリ空間の1つは上記複数のCPUの各1つのために用いられることを特徴とする請求項28記載のシステム。

(30) 上記複数のCPUによる上記共通メモリへのメモリアクセスは、実行される前に、上記共通メモリによってポートされることを特徴とする請求項28記載のシステム。

(31) 上記複数のCPUによる上記プライベートメモリへのメモリアクセスは、データではなくアドレスを比較してポートされることを特徴とする請求項30記載のシステム。

(32) 上記各CPUのための上記プライベートメモリは、上記複数のCPUによって実行される命令と関連する同一の論理アドレスを有するが、上記共通メモリに対してアドレス指定する前に上記各プライベートメモリのための唯一のアドレス

における同一の命令の実行時において行われることを特徴とする請求項24記載の方法。

(26) 上記複数のCPUは、すべての上記CPUによってアクセス可能である共通のメモリへのアクセスをポートすることによって緩く同期化されることを特徴とする請求項24記載の方法。

(27) 3個の上記CPUが存在し、上記3個のCPUは2重化された共通のグローバルメモリをアクセスすることを特徴とする請求項24記載の方法。

(28) 同一の命令ストリームを実行するマルチCPUと、

すべての上記CPUによってアクセスされるメモリ空間を有する共通メモリと、

1個のCPUによってのみ書き込み可能な上記各CPUのための状態情報を格納するための上記共通メモリにおけるプライベートメモリ空間と、

すべての上記CPUによって読み出し可能であるすべての上記CPUのための上記プライベートメモリ空間における上記状態情報とを備え、これ

に翻訳されることを特徴とする請求項28記載のシステム。

(33) 多重CPUを有するコンピュータシステムであって、

すべての上記多重CPUによってアクセスされるメモリ空間を有する分割されたメモリと、

上記多重CPUの各1つはまた状態情報を格納するための上記分割されたメモリにおいて独立したプライベートライトメモリ空間を有し、上記各プライベートライトメモリ空間は上記多重CPUの1つによってのみ書き込み可能であり、

上記マルチCPUの各1つのための上記プライベートライトメモリ空間は、すべての上記多重CPUによって読み出し可能であることを特徴とするコンピュータシステム。

(34) 上記多重CPUは同一の命令ストリームを実行することを特徴とする請求項33記載のシステム。

(35) 上記分割されたメモリは、上記多重CPUによる上記分割されたメモリへ行われるメモリ

要求をポートすることを特徴とする請求項34記載のシステム。

(36) 上記分割されたメモリは、データではなくアドレスを比較することによって、上記プライベートメモリライト空間に対してなされるライト要求をポートすることを特徴とする請求項33記載のシステム。

(37) 多重プロセッサを有するコンピュータシステムを動作させるための方法であって、

すべての上記多重プロセッサによってアクセスされるメモリ空間を有する分割されたメモリに上記各多重プロセッサによってデータを格納するステップと、

1個の多重プロセッサによってのみ書き込み可能である各多重プロセッサのためのプライベートメモリ空間に上記多重プロセッサの各1つによって情報をまた格納するステップとを備えたことを特徴とする方法。

(38) 上記多重プロセッサの各1つにおいて同一の命令ストリームを実行するステップを含む請

上記プライベートメモリ空間における上記情報を読み出すステップを含むことを特徴とする請求項37記載の方法。

(43) 上記方法は、上記多重プロセッサの各1つにおいて同一の命令ストリームを実行するステップを含み、上記データを格納するステップは、上記多重プロセッサによって行われる上記分割されたメモリへのメモリ要求をポートすることを含むことを特徴とする請求項42記載の方法。

(44) 上記多重プロセッサは、メモリ要求をポートする場合に、緩く同期化されることを特徴とする請求項43記載の方法。

### 3. 発明の詳細な説明

(産業上の利用分野)

本発明は、コンピュータシステムに関し、より詳しくは、多重CPUを有するフォールトトレラント(fault tolerant)コンピュータにおいて用いられるメモリ管理システムに関する。

(従来技術と発明が解決しようとする課題)

高信頼性のデジタル処理、冗長性を用いた様々

な請求項37記載の方法。

(39) 上記データを格納するステップは、上記多重プロセッサによってなされる上記分割されたメモリへのメモリ要求をポートすることを特徴とする請求項37記載の方法。

(40) プライベートメモリ空間に情報を格納するステップは、上記各多重プロセッサによってすべての上記プライベートメモリ空間に対するライト要求を行うことを含むが、上記各プライベートメモリ空間と関連する各ライト要求に対して1個のプロセッサに対してのみライト要求を実行することを含むことを特徴とする請求項37記載の方法。

(41) 上記方法はさらに、上記多重プロセッサの各1つによって上記プライベートメモリ空間から上記情報に対して等しいか否かの評価を行うステップを含むことを特徴とする請求項37記載の方法。

(42) 上記方法はさらに、上記各多重プロセッサによってすべての上記多重プロセッサのために

なコンピュータアーキテクチャにおいて達成される。例えば、TMR(3重・モジュラ・冗長性)システムは、同じ命令のストリーム(流れ)を実行する3個のCPUを、機能を重複する3個の分離した主メモリユニットと分離した入出力(以下、I/Oという。)装置とともに使用できる。そのため、もし各タイプの要素の中の1つが誤りをしても、システムは動作し続ける。他のフォールトトレラントタイプのシステムが、カップマン等に対して発行されタンドム・コンピュータズ・インコーポレイテッドに対して譲渡された「多重プロセッサシステム」と題する米国特許第4,228,496号に示される。様々な方法が、冗長性システムにおいて装置を同期させるために使用されて来た。例えば、「多重プロセッサを同期させるための方法と装置」と題するアール・ダブリュ・ク・ホーストにより1987年11月9日に出願され、同様にタンドム・コンピュータズ・インコーポレイテッドに譲渡された米国特許出願第18,503号において、「緩い」同期法が開示

されているが、これは、「フォールトトレラント計算のための中央処理装置」と題されストラス・コンピュータ・インコーポレイテッドに譲渡された米国特許第4,453,215号において示されているような単独のクロックを用いたロック・ステップ同期を使用した他のシステムと対照的である。「同期ボートイング (synchronization voting)」と呼ばれる技法がデビス (Davies) 及びウエイカリ (Wakerly) 著「冗長性システムにおける同期とマッチング」(IEEEトランザクションズ・オン・コンピュータ (IEEE Transactions on computer), 1978年6月号531-539ページ) に開示されている。冗長性のフォールトトレラントシステムにおける割り込み同期の方法が、ヨンディ (Yondea) ほか著「長く同期したTMRシステムのための割り込み取り扱いの実行」(フォールトトレラント計算についての第15回年次シンポジウムのプロシーディング (1985年6月) 246-251ページ) に開示されている。「フォールトトレラントリアル

テムの分野でもなされ、すなわち、標準的オペレーティングシステムを利用できる必要がある。

したがって、この発明の主目的は、特にフォールトトレラントタイプの改良された高信頼性コンピュータシステムを提供することである。この発明の他の目的は、改良された冗長性でフォールトトレラントタイプのコンピュータシステムであって、高性能と低コストが両立するものを提供することである。特に、改良されたシステムが、高度に冗長なシステムにおいて通常生じる実行負荷を避けることが好ましい。この発明の別の目的は、速度とソフトウェアの両立性とともに信頼性について測定される場合に、性能が改良されている一方、コストも他のより低い性能のコンピュータシステムと同じぐらいである高信頼性コンピュータシステムを提供することである。この発明のさらに他の目的は、デマンドページングを用いた仮想メモリ管理を使用し、保護された(上位からの監視、すなわち「核(カーネル: kernel)」)モードを備えたオペレーティングシステムを実行でき

タイムクロック」と題する米国特許第4,644,498号は、TMRコンピュータシステムにおける使用のための3重モジュラ冗長性クロック成を開示している。「多重に冗長なコンピュータのフレーム同期」と題する米国特許第4,733,353号は、同期フレームを実行することにより周期的に同期される別々のクロックで動作するCPUを用いる同期法を開示している。

25MHzで動作するインテル80386やマイクロラ68030のような高性能マイクロプロセッサ装置が、高速クロックと大きな能力を備えて使用できるようになった。また、メモリ、ディスクドライブなどのコンピュータシステムの他の要素もこれに対応してより安価にかつより大きな能力を備えるようになった。このため、高信頼性のプロセッサが同じ傾向に追随することが要求されている。さらに、コンピュータ産業におけるいくつかのオペレーティングシステムでの標準化は、アプリケーションソフトウェアの利用性を大きく拡大した。そのため、同様な要求が高信頼性シス

テの高信頼性コンピュータシステムを提供することである。とくに、オペレーティングシステムは、多重プロセスの実行が、すべて高レベルの性能で可能でなければならない。この発明のさらに別の目的は、故障システム部品を検出できオフラインでそれらを交換でき、システムをダウンさせることなく補修されたシステム部品を再統合できる高信頼性冗長性コンピュータシステムを提供することである。

(課題を解決するための手段、作用及び発明の効果)

この発明の一実施例によれば、コンピュータシステムは、典型的には同じ命令ストリームを実行する3個の同一のCPUを使用し、同じデータの複製を格納する2個の同一の自己診断メモリモジュールを備える。したがって、古典的TMRシステムにおけるような3個のCPUと3個のメモリよりはむしろ、3個のCPUと2個のメモリの構成が使用される。3個のCPUによるメモリ参照(memory reference)は、2個のメモリの各の3個

の別のポートに接続された3個のバスにより行われる。フォールトトレラント動作の実行負荷を全CPU自身に課することを避けるため、また、フォールトトレラントクロック動作の費用、復さ及びタイミングの問題を課することを避けるため、3個のCPUはそれぞれ、それ自身のために独立したクロックを別々に備えるが、メモリ参照のようなイベント(event)を検出することにより、すべてのCPUが、同時に機能を実行するまで他のCPUの前にある任意のCPUをストールすることにより、緩く同期されている。割り込みもまた、全CPUに同期され、全CPUが命令ストリームの同じ点で割り込みを実行することを保証する。別々のCPU-メモリ・バスを介しての3個の非同期のメモリ参照は、メモリ要求のときに各メモリモジュールの3個の別々のポートでポートされるが、リードデータは、全CPUに戻されたときにポートされない。

2個のメモリは、共に、全CPUまたは全I/O(すなわち入力/出力)バスから受け取ったす

ポートされないリードデータの戻りと重なったアクセスの特徴は、高性能のフォールトトレラント動作を、最小の複雑さと費用で可能にする。

I/O機能は、2つの同一のI/Oバス(各バスはただ1個のメモリモジュールと別々に接続される)を用いて実行される。多数のI/Oプロセッサが2つのI/Oバスに接続され、I/O装置は、複数の対のI/Oプロセッサに接続されるが、ただ1個のI/Oプロセッサによってアクセスされる。1個のメモリモジュールがプライマリとして表されるので、このモジュールのためのただ1個のI/Oバスが、全I/Oプロセッサを制御する。そして、メモリモジュールとI/Oとの間のトラフィックは、ポート(vote)されない。全CPUは全I/Oプロセッサをメモリモジュールを介してアクセスできる。(ここで、各アクセスは、まさにメモリアクセスがポートされるようにポートされる。)しかし、全I/Oプロセッサは、全メモリモジュールをアクセスできるだけであり、全CPUをアクセスできない。全I/Oプロセッサ

すべてのライト要求を実行するので、両メモリは、最新に保たれる。しかし、ただ1個のメモリモジュールは、リード要求に対応して全CPUまたはI/Oバスに戻る。リードデータを作る1個のメモリモジュールが「プライマリ」(「主」と呼ばれ、他方はバックアップである。従って、入ってくるデータは、ただ1つのソースからであり、ポートされない。2個のメモリモジュールへのメモリ要求は、ポート航行中は実行されるが、従って、リードデータは、最後のCPUが要求を行った後で少し遅れて全CPUに対し利用できる。これらのメモリモジュールのために使用されるDRAMが単にリード動作を行いリフレッシュするためにライトサイクルの大部分を使用するので、ライトサイクルでさえも実質的に重複し得る。そこで、ライトサイクルの最後の部分のためにストロープされないならば、リード動作は非破壊的でない。従って、ライトサイクルは、最初のCPUが要求をすると直ちに開始されるが、最後の要求が受信され、良好であるとポートされるまで完了しない。

は、全CPUに割り込みを送ることができるだけであり、この割り込みは、全CPUに示される前にメモリモジュール内に集められる。こうして、I/O装置アクセスのための同期オーバーヘッドは、全CPUにとって重荷にならず、フォールトトレラント性が備えられる。もし1個のI/Oプロセッサが誤ったならば、その対の他方のI/Oプロセッサが、オペレーティングシステムにより維持されるI/Oページテーブル内のI/O装置に対して用いられるアドレスを単に変えるだけで、このI/OプロセッサのためのI/O装置の制御を代わって行うことができる。このように、I/O装置のフォールトトレラント性と再統合は、システムシャットダウンなしに、そしてさらに、これらのI/Oバスにおけるポーティングに伴うハードウェア費用と実行ペナルティなしに可能である。

説明された実施例において使用されるメモリシステムは、複数のレベルで階層的である。各CPUは、それ自身のキャッシュ(cache)を備え、本質的にCPUのクロック速度で動作する。そこで、

各CPUは、他のCPUによりアクセスできないローカルメモリを備え、仮想メモリは、オペレーティングシステムの核と現在のタスクのページを全3個のCPUのためのローカルメモリの中にあることを許可し、課されたポーティングまたは同期のようなフォールトトレラント性のオーバーヘッドなしに高速でアクセス可能にする。次に、グローバルメモリとして呼ばれるメモリモジュールレベルがあり、ここで、ポーティングと同期化が行われ、アクセスタイムの負荷が導入される。しかし、グローバルメモリの速度は、ディスクアクセスよりもずっと速い。従って、このレベルは、デマンドページングの第1レベルのためにディスクを使用するためよりはむしろ、最速のエリアに最も使用されるデータを保持するためのローカルメモリとの、ページのスワッピングのために使用される。

この発明の開示された実施例の1つの特徴は、システムをシャットダウンすることなしにCPUモジュールやメモリモジュールのような故障部品

能なフォールトトレラントシステムは、一時的に広く使用されるマルチタスクのオペレーティングシステムとアプリケーションソフトウェアとの同等性を可能にして提供される。

メモリモジュールは本質的には2重化され、又は互いに同一のデータを格納するが、データがすべてのCPUによって読み出し可能であるような方法で、各CPUによって別々にデータを格納することを可能にするための必要性がいくつかの状態においてまだ存在する。もちろん、例示の実施例の複数のCPUは（メモリモジュールのみならず、CPUモジュールの代替において）ローカルメモリを有するが、このローカルメモリを他のCPUによってアクセスすることはできない。このように、本発明の1つの特徴によれば、プライベートライトメモリの領域は、唯一の状態情報を各CPUによって書き込むことができ、次いで、他のCPUによって読み出し、例えば比較動作を行うことができるように、その分割されたメモリ領域において含まれる。このプライベートな

を交換する能力である。こうして、このシステムは、部品が故障し、取り換えねばならない場合でさえも、連続的な使用ができる。さらに、高レベルのフォールトトレラント性がより少ない部品で達成できる。例えば、フォールトトレラントなクロック動作が必要でなく、3個でなく2個のメモリモジュールだけが必要であり、ポーティング回路が最小にできる。このことは、故障する部品が少なく、信頼性が増大したことを意味する。すなわち、部品がより少ないので、故障がより少なく、故障があるとき、システムをランさせたまま、その部品が分離され、システムシャットダウンなしに取り換えてできる。

このシステムのCPUは、好ましくは、UNIX（登録商標）のようなオペレーティングシステムが使用可能な市販の高性能マイクロプロセッサチップを使用する。システムをフォールトトレラントにする部分は、オペレーティングシステムに対して透明であるか、またはオペレーティングシステムに対して容易に適合できる。従って、高性

書き込みは、複数のCPUの命令ストリームがまだ同一であり、用いられるアドレスが同一であるような方法でアクセスされ、その結果、同一のコードストリームの完全な状態が維持される。プライベートな書き込み動作が複数のメモリモジュールによって検出されたとき、これらのデータは異なる可能性があるので、データのポーティングは一時中止されるが、アドレスとコマンドはいまだポートされる。プライベートな書き込みのために用いられる領域を、命令ストリームの制御のもとで、変化し又は除去してもよい。従って、唯一のデータを比較するための能力は、同期化をバイパスせずかつ機構をポートすることなしに、かつ多重CPUによって実行されるコードの同一の特徴を乱すことなしに、柔軟性のある方法で提供される。

（以下余白）

## (実施例)

以下、添付の図面を参照して本発明の実施例を説明する。

第1図を参照して、本発明の特徴を用いたコンピュータシステムは、一実施例において、論理プロセッサとして動作する3個の同一のプロセッサ11、12及び13（以下、それぞれCPU-A、CPU-B及びCPU-Cという。）を備え、これら3個は、典型的には同じ命令ストリームを実行する。3個のプロセッサが同じ命令ストリームを実行しない唯一の時間は、システム起動自己テスト、診断などの動作である。3個のプロセッサは、2個のメモリモジュール14と15（メモリ#1、メモリ#2と呼ばれる）と接続され、各メモリは、同じアドレス空間に同一のデータを格納する。好ましい実施例においては、各プロセッサ11、12及び13は、その固有のローカルメモリ16を含み、このメモリを含むプロセッサによってのみアクセス可能である。

各プロセッサ11、12及び13は、各メモリ

時に同期化され、その結果、プロセッサは、典型的には、同じ命令ストリームを、同じシーケンスで、ただし必ずしも同期イベントの間の時間における平行した時間サイクルの間ではないが、実行する。さらに、外部の割り込みは、同期化されて、各CPUの命令ストリームにおける同一の点で実行される。

CPU-Aプロセッサ11は、バス21を介して、メモリ#1モジュール14とメモリ#2モジュール15に接続される。同様に、CPU-Bプロセッサ12は、バス22を介して、メモリ#1モジュール14とメモリ#2モジュール15に接続される。そして、CPU-Cプロセッサ13は、バス23を介して、メモリモジュール14、15に接続される。これらのバス21、22、23は、32ビット多重アドレス/データバス、コマンドバス、及びアドレスとデータのストロブのための制御ラインを含む。これらのCPUは、これらのバス21、22及び23の制御を備え、そのため、アービトレーション (arbitration) または

モジュール14と15と同様に、それ自身の固有の別々のクロック発振器17を備える。この実施例において、全プロセッサは、「ロックステップ」でランされず、その代わり、上述の米国出願第118,503号で明らかにされたような方法により、すなわち、これらのCPUを同期化させる外部メモリ参照のようなイベントを使用して、緩く同期される。外部の割り込みは、各プロセッサから他の2個のプロセッサへ割り込み要求とステータスを結合するための1組のバスを使用する技法によって、3個のCPUの間で同期化される。各プロセッサCPU-A、CPU-B及びCPU-Cは、それ自身と他の2個との3個の割り込み要求に対して応答的であり、命令ストリームの同じ点においてこれらのCPUに割り込み要求を示す。メモリモジュール14と15は、メモリ参照をポートし、全3個のCPUが同じ要求（故障に対する準備とともに）を行ったときにのみ、メモリ参照が連むことを許可する。このように、これらのプロセッサは、外部のイベント（メモリ参照）の

バス要求やバス使用承認 (bus grant) はない。

各メモリモジュール14と15は、それぞれの入出力バス24又は25に別々に接続され、各バスは、2個（またはそれ以上）の入出力プロセッサに接続される。このシステムは、個々のシステム構成のために必要なI/O装置を収容するために必要な多数のI/Oプロセッサを備えることができる。各入出力プロセッサ26、27は、バス28に接続される。バス28は、VMEバス（登録商標）のような標準の構成であってもよい。そして、各バス28は、標準のI/Oコントローラ30とのインターフェースのための1個以上のバスインターフェースモジュール (BIM) 29に接続されている。各バスインターフェースモジュール29は、2個のバス28に接続され、従って、1個のI/Oプロセッサ26または27の故障、または1個のバスチャンネル28の故障は、許容される。I/Oプロセッサ26と27を、CPU11、12及び13によってメモリモジュール14と15を通してアドレス指定することができ、

I/Oプロセッサ26、27はメモリモジュールを介して全CPUに割り込み信号を出力することができる。ディスクドライブ、CRTスクリーンとキーボードを備えたターミナル、及びネットワークアダプタは、I/Oコントローラ30により作動される典型的な周辺装置である。I/Oコントローラ30は、データブロックのような転送のためにメモリモジュール14と15に対しDMAタイプの参照をすることができる。各I/Oプロセッサ26、27などは、バス要求、バス使用承認等のために各メモリモジュールに直接に接続された個々のラインを備える。これらの点から点への接続ラインは、「ラジアル」と呼ばれ、ラジアルライン31のグループに含まれる。

システムステータスバス32は、各素子のステータス情報を与える目的のために、上記各CPU11、12、13、各メモリモジュール14、15、各I/Oプロセッサ26、27に、個々に接続される。このステータスバスは、システムに現在存在し適当に動作しているCPU、メモリモジュール

メモリに同じデータを格納し、すべてのメモリ参照を2重に行うように動作しているが、任意の与えられた時間では、1個のメモリモジュールがプライマリと指定され、他方は、バックアップと指定される。メモリライト動作は、両メモリモジュールにより実行されるので、両方とも使用可能状態(current)であり、またメモリリード動作も両方により実行される。しかし、プライマリのメモリモジュールのみが、バス21、22及び23に実際にリードデータをロードし、そして、プライマリのメモリモジュールのみがマルチマスタバス24と25のためのアービトレーションを制御する。プライマリのメモリモジュールとバックアップのメモリモジュールに同じ動作の実行を統括するために、バス34がプライマリからバックアップへ制御情報を伝送する。どちらかのメモリモジュールが、ブートアップにおいてプライマリの役割を取り、この役割は、ソフトウェアの制御の下に動作の間に交換できる。当該役割は、選択されたエラー条件が全CPUまたはシステムの他のエラ

ール及びI/Oプロセッサについての 報を提供する。

3個のCPUと2個のメモリモジュールを接続する肯定応答/ステータスバス33は、メモリ要求が全CPUによって行われたときにモジュール14、15が全CPUに肯定応答信号を送信する個々のラインを含む。同時に、ステータスフィールドが、コマンドのステータスとコマンドが正しく実行されたか否かについて報告するために送信される。メモリモジュールは、グローバルメモリから読み出されたデータまたは書き込まれたデータのパリティを検査するだけでなく、メモリモジュールを介してI/Oバス24と25へまたはバス24、25からのデータのパリティを検査し、またコマンドの正当性を検査する。これらの検査がCPU11、12及び13に報告されるのは、バス33のステータスラインを介してであり、もし誤りが発生すると、故障ルーチンを、故障部品を分離するためにエンターすることができる。

2個のメモリモジュール14と15がグローバ

一応答性部分によって検出されるときに、交換できる。

全CPUにおいて発生されたある割り込みは、また、メモリモジュール14と15によってポートされる。全CPUがそのような割り込み状態となったとき(及びストールされないとき)、全CPUは割り込みバス35の個々のラインによって全メモリモジュールに割り込み要求を出力する。そこで、3個のCPUからの3個の割り込み要求をポートすることができる。すべての割り込みがポートされたとき、メモリモジュールは、それぞれバス35を介して3個のCPUにポートされた割り込み要求信号を送信する。この割り込みのポーティングは、また、全CPUの動作についての検査のために機能する。3個のCPUは、CPU間バス18を介してこのポートされた割り込みをCPU割り込み信号に同期し、命令ストリームの共通の点で全プロセッサに割り込みを示す。この割り込み同期は、どのCPUもストールせずに達成される。

## ＜CPUモジュール＞

第2図を参照して、1個のプロセッサ11、12又は13がさらに詳細に示される。全3個のCPUモジュールは、好ましい実施例では、同じ構成であり、従って、CPU-Aのみがここで説明される。価格を競争力のある範囲内に保つために、そして、既に発展されているソフトウェアとオペレーティングシステムへのアクセスをただちに提供するために、好ましくは、市販のマイクロプロセッサチップが使用され、多数のデバイスの中の任意の1個が選択できる。RISC（縮小命令セット）アーキテクチャは、後述する緩い同期を実行することにおいて利点がある。しかし、モトローラ68030デバイスやインテル80386デバイス（20MHzと25MHzで使用できる）などのより通常のCISC（複雑な命令セット）マイクロプロセッサが使用できる。高速32ビットRISCマイクロプロセッサデバイスは、3個の基本的なタイプで多数の製造者から入手できる。すなわち、モトローラは、部品番号88000と

デバイス・テクノロジー・インコーポレイテッドによって製造される。このR2000デバイスは、RISCアーキテクチャを用いた32ビットプロセッサであり、例えば、16.67MHzのクロックで12MIPSの高性能を示す。25MHzのクロックで20MIPSを示すR3000のようなこのデバイスのより高速のバージョンに用いても良い。プロセッサ40はまた、論理アドレスから物理アドレスへの翻訳をキャッシュするためのトランслーションルックアサイドバッファを含むメモリ管理のために使用されるコプロセッサを備える。プロセッサ40は、データバス、アドレスバス、および制御バスを備えたローカルバスに接続される。別々の命令とデータのキャッシュメモリ44と45が、このローカルバスに接続される。これらのキャッシュは、それぞれ64Kバイトサイズであり、プロセッサ40の1つのクロックサイクル内でアクセスされる。もし追加の性能がこれらのタイプの計算のために必要ならば、数値計算用すなわち浮動小数点コプロセッサ46が、

してデバイスを製造し、MIPSコンピュータ・システムズ・インコーポレイテッドなどは、MIPSタイプと呼ばれるチップセットを製造し、サン・マイクロシステムズは、いわゆるSPARC（登録商標）タイプ（スケール可能なプロセッサアーキテクチャ）を発売している。カリフォルニア州サンホセのサイプレス・セミコンダクタは、例えば、部品番号CY7C601と呼ばれるマイクロプロセッサ（SPARC標準をサポートし、33MHzのクロックを用い、20MIPSの（1秒当たり100万命令）を与える）を製造し、富士通は、同様にSPARC標準をサポートするCMOS RISCマイクロプロセッサ（部品番号S-25）を製造している。

図示された実施例におけるCPUボードすなわちモジュールは、一例として使用され、マイクロプロセッサチップ40を用いる。このチップ40は、この場合MIPSコンピュータ・システムズ・インコーポレイテッドにより設計されたR2000デバイスであり、また、インテグレイテッド・

このローカルバスに接続される。この数値計算用プロセッサデバイスも、MIPSコンピュータ・システムズ・インコーポレイテッドから部品番号R2010として市販されている。ローカルバス41、42、43は、ライトバッファ50とリードバッファ51を介して内部バス構造に接続される。このライトバッファは、入手可能なデバイス（部品番号R2020）であり、ライト動作のためにライトバッファ50にデータとアドレスを格納した後に、ライトが実行されている間にストローサイクルを実行しなければならないことよりはむしろ、プロセッサ40にラン（Run）サイクルを実行し続けさせるように機能する。

ライトバッファ50を通るバスに加え、プロセッサ40がライトバッファ50をバイパスしてライト動作を実行することを可能にするためのバスが設けられる。このバスは、ソフトウェアの選択の下で、プロセッサに同期のライト動作を行うことを可能にする。もしライトバッファバイパス52がイネーブルされ（ライトバッファ50がイネー

ブルされず)、プロセッサがライト動作を実行するならば、プロセッサは、ライト動作が完了するまでストール(一時停止)する。対照的に、ライトバッファがディスエーブルの状態ではライト動作が実行されるとき、データがライトバッファ50に書き込まれるので(ライトバッファが満杯でないならば)、プロセッサはストールしない。もしプロセッサ40がライト動作を実行するときにライトバッファがイネーブルされるならば、ライトバッファ50は、バス43からの制御と同様に、バス41からの出力データとバス42からのアドレスを捕獲する。ライトバッファ50は、主メモリへのデータの通過を待機する間に最大4個のそのようなデータ-アドレスセットを保持できる。ライトバッファはプロセッサチップ40のクロック17と同期して動作し、このため、プロセッサからバッファへの転送は同期状態でかつプロセッサのマシンサイクル速度で行われる。ライトバッファ50は、もし満杯であってデータを収容できないならば、プロセッサに信号を送信する。プロ

セッサ40によるリード動作は、フォーディーブ・

ライトバッファ50に含まれるアドレスに対して検査され、そこで、もしメモリ16すなわちグローバルメモリに書き込まれるためにライトバッファで待機しているデータに対してリード動作が試みられるならば、リード動作は、ライト動作が完了するまでストールされる。

ライトバッファ50とリードバッファ51は、データバス53、アドレスバス54および制御バス55を備えた内部バス構造に接続される。ローカルメモリ16は、この内部バスによってアクセスされ、この内部バスに接続されたバスインターフェース56は、システムバス21(または他のCPUのためのバス22または23)をアクセスするために使用される。この内部バスの別々のデータバス53とアドレスバス54(ローカルバスのバス41と42から得られる)は、システムバス21内の多重化アドレス/データバス57に変換され、コマンドラインと制御ラインは、対応して、この外部バス内のコマンドライン58と制御

ライン59に変換される。  
バスインターフェースユニット56は、また、メモリモジュール14と15から肯定応答/ステータスライン33を受信する。これらのライン33において、別々のステータスライン33-1または33-2は、モジュール14及び15のそれぞれから接続され、その結果、両メモリモジュールからの応答を、後述するように、複数のCPUとグローバルメモリの間の転送(リードまたはライト)の発生の場合に評価できる。

一実施例においては、ローカルメモリ16は、約8MバイトのRAMからなり、プロセッサ40の約3個または4個のマシンサイクル内でアクセスでき、このアクセスは、このCPUのクロック17と同期している。これに反し、モジュール14と15へのメモリアクセスタイムは、ローカルメモリへのそれに比べて非常に長く、メモリモジュール14、15へのこのアクセスは、非同期であり、すべてのCPUが要求とポーティングとを行うことを待機することにより遅られる同期のオー

バーヘッドをこうむる。比較のため、1/Oプロセッサ26、27、及び29を介しての典型的な市販のディスクメモリへのアクセスは、ミリ秒で測定され、すなわち、モジュール14と15へのアクセスよりもかなり遅い。こうして、CPUチップ40によるメモリアクセスの階層構造がある。最高は、命令キャッシュ44とデータキャッシュ45であり、64Kバイトのキャッシュサイズと適当なフィアルゴリズム(fill algorithm)を使用したときに多分95%のヒット率を示す。最高の次は、ローカルメモリ16であり、再び一時的仮想メモリ管理アルゴリズムを使用することにより、ローカルメモリのサイズが約8Mバイトである場合に、キャッシュミスが発生し、ローカルメモリにおけるヒットが見いだされ、おそらく95%のヒット率が、メモリ参照に対して得られる。プロセッサチップの観点からの正味の結果は、メモリ参照(1/O参照でなく)のおそらく99%以上が同期し、同じマシンサイクルまたは3個または4個のマシンサイクル内に起こることである。

ローカルメモリ16は、メモリコントローラ60によって内部バスからアクセスされる。このメモリコントローラ60は、アドレスバス54からのアドレスと制御バス55からのアドレスストローブを受信し、例えば、もしローカルメモリ16が通常のように多重アドレス指定でDRAMを使用するならば、別々の行と列のアドレスと、RASとCASの制御を発生する。データは、データバス53を介してローカルメモリに書き込まれ、読み出される。さらに、オペレーティングシステムによって使用可能なので、NVRAMや高速PROMのような不揮発性メモリ62と同様に、数個のローカルレジスタ61が、内部バスによってアクセスされる。メモリのこの部分のいくつかは電源投入のためにのみ使用され、いくつかはオペレーティングシステムによって使用され、キャッシュ44内でほとんど連続的であり、他は、メモリマップのキャッシュでない部分内に有り得る。

外部割り込みは、第2図のCPUモジュールの割り込み回路65から制御バス43または55で

外部ソースにて発生された割り込みは、例えば、各CPU11、12又は13が、後述されるように、命令ストリーム内の同じ点にあるまで、回路65からチップ40の割り込みピンに印加されない。

プロセッサ40は別々のクロック発生器17によってクロックが供給されるので、周期的にプロセッサ40を同期状態に戻すためのいくつかのメカニズムが必要である。クロック発生器17が名目上同じ周波数でありこれらのデバイスの許容誤差が約25ppm (parts per million) であったとしても、これらのプロセッサは、周期的に同期に戻されないならば、位相が多くのサイクルでずれてしまう可能性がある。もちろん、外部割り込みが発生する毎に、全CPUは、(割り込み同期メカニズムによって) その命令ストリームの同じ点で割り込まれるという意味で、同期化される。しかし、これは、サイクル計数値を同期化させることを要しない。メモリモジュール14と15内のメモリ参照をポートするメカニズムは、後述

される。このタイプの割り込みは、回路65でポートされるので、割り込みがプロセッサ40によって実行される前に、全3個のCPUが割り込みを示されるか否かが決定される。この目的のために、回路65は、他の2個のCPU12と13から割り込み未決定(pending)入力を受信し、この他の2個のCPUにライン67を介して割り込み未決定信号を送信する。これらのラインは、3個のCPU11、12及び13をとともに接続するバス18の一部である。また、他のタイプの割り込み例えばCPUにより発生された割り込みをポートするために、回路65は、このCPUから両メモリモジュール14、15へバス35のライン68により割り込み要求信号を送信することができ、そして、ライン69と70を介してメモリモジュールから別々のポートされた割り込み信号を受信する。両メモリモジュールは、行われるべき外部割り込みを与える。1個のI/Oチャンネル28でのキーボードまたはディスクドライブのような

されるように全CPUをリアルタイムで同期状態にする。しかし、ある条件は、長い周期においてメモリ参照が起こらないという結果を生じ、そこで、別のメカニズムが、プロセッサ40を同期に戻すためのストールサイクルを導入するために使用される。サイクルカウンタ71は、ランサイクル(ストールサイクルでなく)であるマシンサイクルを計数するために、クロック17とプロセッサ40の制御ピンに制御バス43を介して接続される。このカウンタ71は、全CPUの間の最大の許容可能なドリフトが発生する周期(結晶発振子の特定の許容誤差を考慮して)を表すように選択された最大計数値を有するカウントレジスタを含む。このカウントレジスタがオーバーフローすると、より遅いプロセッサが追い付くまで、より速いプロセッサをストールする動作が開始される。このカウンタ71は、メモリモジュール14と15へのメモリ参照によって同期がなされるときはいつでもリセットされる。また、リフレッシュカウンタ72は、後述されるように、ローカルメモ

リ16でリフレッシュサイクルを実行するために使用される。さらに、カウンタ73は、カウンタ71のように、ランサイクルであってストールサイクルでないマシンサイクルを計数する。しかし、このカウンタ73は、メモリ参照によってリセットされない。カウンタ73は、以下に説明されるように、割り込み同期のために使用され、この目的のために、割り込み同期回路65に出力信号C C-4とC C-8を発生する。

プロセッサ40は、RISC命令セットを備え、このセットは、メモリからメモリへの命令をサポートしないが、その代わり、メモリからレジスタへの命令またはレジスタからメモリへの命令（たとえばロードまたはストア）をサポートする。ローカルメモリにしばしば使用されるデータや現在実行中のコードを保持することは重要である。従って、ブロック転送動作は、バスインターフェース56に結合されたDMAステートマシン74によりなされる。プロセッサ40は、コマンドとして機能させるためにDMA回路74のレジスタに1

ワードを書き込み、この回路74のレジスタにブロックのスタートアドレスと長さを書き込む。一実施例では、DMA回路がブロック転送を引き起こす間に、マイクロプロセッサはストールをして、バス53-55及び21によって必要なアドレス、コマンド及びストローブを発生する。このブロック転送を開始するためにプロセッサ40によって実行されるコマンドは、DMA回路74のレジスタからのリードであってもよい。UNIXオペレーティングシステムにおけるメモリ管理はデマンドページングを当てにしているので、これらのブロック転送は、最もしばしばグローバルメモリとローカルメモリとI/Oトラフィックの間に動かされるページである。1ページは4Kバイトである。もちろん、バス21、22及び23は、CPUとグローバルメモリの間の1ワードのリード転送とライト転送をサポートする。参照されるブロック転送は、ローカルメモリとグローバルのメモリの間でのみ可能である。

<プロセッサ>

第3図を参照して、実施例のR2000タイプまたはR3000タイプのプロセッサ40がさらに詳細に示される。このデバイスは、32個の32ビットの一般目的のレジスタ76、32ビットのALU77、0ビットから64ビットへのシフタ78、および32×32の多重/分割回路79を備える32ビットのメインCPU75を備える。このCPUは、また、プロセッサバス構造81に接続され、このプロセッサバス構造81は、ローカルデータバス41に接続され、データバス41を介してフェッチされる命令を実行するための関連する制御ロジックを備えた命令デコード82に接続される。32ビットのローカルアドレスバス42は、オンチップメモリ管理コプロセッサ内のトランスレーションルックアサイドバッファ（TLB）83を含む仮想メモリ管理装置によって駆動される。TLB83は、仮想アドレスバス84を介してマイクロプロセッサブロック75から受け取られた仮想アドレスと比較されるべき64個のエントリを備える。バス42の下位の16ビッ

トの部分85は、この仮想アドレスバス84の下位部分によって駆動され、上位部分は、もし仮想アドレスが物理的地址として使用されるならば、バス84からであり、あるいは、もし仮想アドレス指定が使用され、ヒットが起こるならば、出力86を介してのTLB83からのタグエントリである。ローカルバスの制御ライン43は、パイプライン及びバス制御回路87に接続され、内部バス構造81と制御ロジック82から駆動される。

プロセッサ40のマイクロプロセッサブロック75は、RISCタイプであり、多くの命令が1マシンサイクルで実行され、命令セットは、ALU動作に伴うメモリ参照を含む複雑な命令を含むよりはむしろ、レジスタからレジスタへの命令やロード/ストア命令を使用する。複雑なアドレス指定スキーム（例えば、レジスタA1とレジスタA2の内容の和であるアドレスのオペランドを、レジスタBの内容によりアドレスされる主メモリの位置に見いだされるアドレスのオペランドに加

え、レジスタCに見いだされるアドレスの位置に主メモリにその和の結果をストアせよ。)は、命令セットの一部として含まれない。その代わり、この動作は、次の多数の単純なレジスタからレジスタへの命令やロード/ストア命令にてなされる。すなわち、レジスタA2をレジスタA1に加算せよ、レジスタB内のアドレスのメモリ位置からレジスタB1をロードせよ、レジスタA1とレジスタB1を加算せよ、レジスタCによりアドレスされたメモリ位置にレジスタB1をストアせよ。

コンパイラ技法は、32個のレジスタ76の使用を最大にするために使用され、すなわち、大部分の動作が既にレジスタセットにあるオペランドを見いだすことを保証する。ロード命令は、実際に、1マシンサイクルより長くかかる。このため1命令の潜在(latency)が導入される。ロード命令によってフェッチされるデータは、第2サイクルまで使用されず、もし可能ならば、その間に入るサイクルが、ある他の命令のために使用される。

その間に入るパイプステージにある。

#### <メモリモジュール>

第6図を参照して、1個のメモリモジュール14または15が詳細に示される。両メモリモジュールは、好ましい実施例において、同じ構成であるので、メモリ#1モジュールのみが示される。メモリモジュールは、それぞれ、CPU1、12、13から来る3個のバス21、22、23に接続される3個の入力/出力ポート91、92、93を含む。これらのポートへの入力は、レジスタ94、95、96にラッチされ、各ラッチは、データ、アドレス、コマンド、及びライト動作のためのストロブ、または、アドレス、コマンド、及びリード動作のためのストロブをストアするための別々のセクションを備える。これらの3個のレジスタの内容は、全3個のレジスタのみがすべてのセクションに接続される入力を備えたポート回路100によってポートされる。もし全3個のCPU11、12、13が同じメモリ要求(同じアドレス、同じコマンド)を行うならば(全CP

メインCPU75は、マシンサイクル当たりの命令実行を平均化する目的を容易にするために高度にパイプライン化されている。第4図を参照して、1つの命令が5マシンサイクルを含む周期にわたって実行される。ここで、1マシンサイクルは、16.67MHzのクロック17に対して1クロック周期すなわち60nsecである。この5サイクルすなわちパイプステージは、IF(キャッシュ44からの命令フェッチ)、RD(レジスタセット76からのリードオペランド)、ALU(ALU77での要求される命令を実行)、MEM(もし要求されたならDキャッシュ45をアクセスせよ)、及びWB(ALUの結果をレジスタファイル76に書き)として呼ばれる。第5図からわかるように、これらの5個のパイプステージは、重なっているもので、与えられたマシンサイクル、例えばサイクル5において、命令1#5は、その第1パイプステージすなわちIFパイプステージにあり、命令1#1は、その最後のステージすなわちWBステージにあり、その他の命令は、

Uは典型的には同じ命令ストリームを実行するのでそのような場合がありうる)、メモリ要求は完了することを許容される。しかし、第1メモリ要求が、3個のラッチ94、95、96のいずれかにラッチされると直ちにメモリアクセスを開始するために通過される。この目的のために、アクセス、データ及びコマンドは、データバス101、アドレスバス102およびコマンドバス103を含む内部バスに印加される。この内部バスから、メモリ要求は、アドレスに依存して、そしてシステム構成に依存して様々なリソースにアクセスする。

一実施例において、大きなDRAM104が、メモリコントローラ105を用いて、内部バスによってアクセスされる。このメモリコントローラ105は、アクセスバス102からアドレスと制御バス103からメモリ要求とストロブとを受信し、データ入力とデータ出力がデータバス101に出力されるようにDRAMのための多重の行と列のアドレスを発生する。このDRAM104

はまた、グローバルメモリと呼ばれ、一実施例においては多分32Mバイトのサイズである。さらに、内部バス101-103は、制御・ステータスレジスタ106、多数の不揮発性RAM107及びライトプロテクト108をアクセスできる。CPUによるメモリ参照は、また、メモリモジュール14または15内のメモリをバイパスでき、内部バス101-103に接続される入力を含んだバスインターフェースによってI/Oバス24、25にアクセスできる。もしメモリモジュールがプライマリメモリモジュールであるならば、各メモリモジュール内のバスアービトラクタ110は、バスインターフェース109を制御する。もしメモリモジュールがバックアップモジュールであるならば、バス34はバスインターフェース109を制御する。

DRAM104へのメモリアクセスは、第1の要求が1個のラッチ94、95、又は96にラッチされると直ちに開始されるが、故障に備えて、多数の要求が同じであることをポート回路100

に対するバイトイネーブルとして機能する。上記のストローブは、AS（アドレスストローブ）とDS（データストローブ）である。CPU11、12、13は、それぞれ、自分自身のバス21、22又は23を制御する。この実施例において、これらは、マルチマスタバスではなく、争いすなわちアービトラーションはない。ライトに対して、CPUは、アドレスストローブAS（アクティブでローレベル）で1サイクル内でバスにアドレスとコマンドを送り、続くサイクル（おそらく次のサイクル、しかし必ずしもそうでなくてもよい）でデータストローブと同時にバスのアドレス/データラインにデータを送信する。各CPUからのアドレスストローブASは、ストローブが現れたとき、ポート91、92又は93にアドレスとコマンドを生じさせて、レジスタ94、95、96のアドレス・コマンドセクションにラッチさせ、次に、データストローブDSがデータをラッチさせる。バス21、22、23の多数（この実施例では3の中の2）が同じメモリ要求をラッチ94、

が決定されなければ、完了を許容されない。3個の要求の中の第1の要求の到達は、DRAM104へのアクセスを開始させる。リードに対して、DRAM104がアドレス指定され、センスアンプがストローブされ、データ出力がDRAM入力で生じる。そして、もし第3の要求が受信された後でポートが良いならば、要求されたデータはCPUに直ちに転送するために用意される。このように、ポーティングの動作はDRAMアクセス動作と重なる。

第7図を参照して、バス21、22、23は、図示されたフォーマットにてメモリモジュール14、15のポート91、92、93にメモリ要求を与える。これらのバスの各々は、32本の双方向多重アドレス/データライン、13本の1方向コマンドライン及び2本のストローブからなる。コマンドラインは、リード、ライト、ブロック転送、単独転送、I/OリードまたはI/Oライトなどのバスアクティビティのタイプを特定するフィールドを含む。また、1フィールドは、4バイト

95、96に送信するとき、ポート回路100は、バス103に最後のコマンドを通過させ、メモリアクセスが実行される。もしコマンドがライトならば、ライトが実行されると直ちに、肯定応答ACK信号がライン112（特にメモリ#1のライン112-1とメモリ#2のライン112-2）によって各CPUに送り返され、同時にステータスビットが、第7図の時間T3に各CPUに肯定応答/ステータスバス33（特にメモリ#1のライン33-1とメモリ#2のライン33-2）を介して送信される。最後のストローブDS（もしリードならばAS）とT3でのACKの間の遅延T4は、メモリ要求のときにCPUが何サイクル同期位置からずれているかに依存して、また、ポーティング回路における遅延とCPUクロック17に比べてメモリモジュール14又は15の内部の独立なクロック17の位相に依存して、変わり得る。もしCPUにより出力されるメモリ要求がリードであると、次に、ライン112-1と112-2のACK信号とライン33-1と33-2のステータス

スピットが、時間T3の間に、データがアドレス／データバスに出されるのと同時に送信される。これは、CPUにストールをリリースし、こうして同一の命令に対してCPUチップ40を同期させる。すなわち、最速のCPUは、より速いCPUが追い付くのを待っているため、より多くのストールサイクル(stall cycle)を実行し、こうして、クロック17がたぶん位相がずれているが、全3個が同時にリリースされる。全3個のCPUがストールから出て来たとき、これらのCPUによって最初に行われる命令は同じである。

メモリモジュール14又は15からCPU11、12、13に送信されるすべてのデータは、そのデータがDRAM104から又はメモリ位置106-108からのリードデータであるか、バス24、25からのI/Oデータであるかに拘わらず、レジスタ114を通過する。このレジスタ114は、内部データバス101からロードされ、このレジスタからの出力は、時間T3にポート91、92、93でバス21、22、23のためのアド

レス／データラインに印加される。パリティは、データがこのレジスタにロードされたときに検査される。DRAM104に書き込まれたすべてのデータと、I/Oバスのすべてのデータは、それに関連したパリティビットを持つ。しかし、パリティビットは、バス21、22、23でCPUモジュールに転送されない。リードレジスタ114で検出されたパリティエラーは、ステータスバス33-1、33-2を介してCPUによって報告される。プライマリとして指定されたメモリモジュール14又は15のみが、そのレジスタ114内のデータをバス21、22、23に出力する。バックアップ又はセカンダリとして指定されたメモリモジュールは、レジスタ114をロードする点まで遅延して行われるリード動作とパリティチェックを完了し、バックアップ33-1と33-2にステータスを報告する。しかし、データは、バス21、22、23に出力されない。

各メモリモジュール14又は15内のコントローラ117は、このモジュールのためのクロック

発振器17によりステートマシーンとして動作し、バス103とバス21-23から様々なコマンドラインからのデータ(例えばレジスタとバスをロードするための制御ビットの発生、外部制御信号の発生など)を受信する。現在プライマリとして指定されているモジュール14又は15内のこのコントローラ117は、共通のバス101-103へのアクセスのためにI/O側(インターフェース)とCPU側(ポート91-93)の間でアービトラータ110を介してアービトラートする。プライマリメモリモジュール14又は15のコントローラ117によるこの決定は、ライン34によって他のメモリモジュールのコントローラ117に伝送されて、他のメモリモジュールに同じアクセスを実行させる。

(以下余白)

各メモリモジュール内のコントローラ117はまたこのモジュールのためのクロック発振器17からパルスを受信するリフレッシュカウンタ118に基づいて、DRAM104にリフレッシュサイクルを導入する。DRAMは8ミリ秒毎に512リフレッシュサイクルを受信しなければならず、従って平均して約15ミリ秒毎にリフレッシュサイクルが導入される。こうして、カウンタ118は、15ミリ秒毎にコントローラ117にオーバーフロー信号を発生する。そして、もしアイドル条件(CPUアクセスまたはI/Oアクセスが実行されない)が存在するならば、リフレッシュサイクルがバス103に出力されたコマンドによって実施される。もし動作が進行中ならば、現在の動作が完了したときに、リフレッシュが実行される。メモリのページングに使用されるブロック転送のような長い動作のために、数個のリフレッシュサイクルがバックアップでき、転送が完了した後でバーストモードで実行される。この目的のために、最後のリフレッシュサイクルがカウンタ11

8に関連したレジスタに格納されるので、カウンタ118のオーバーフローの数が用いられる。

CPUによって発生された割り込み要求は、割り込みバス35のライン68によって個々に各CPU11、12、13から受信される。これらの割り込み要求は、各メモリモジュール14、15に送信される。バス35のこれらの要求ライン68は、割り込みポート回路119に接続されて、この回路119はこれらの要求を比較し、バス35の出力ライン69にポートされた割り込み信号を出力する。CPUはそれぞれ、バス35を介して2本のライン（各モジュール14、15から1本）にポートされた割り込み信号を受信する。各メモリモジュール14、15からのポートされた割り込み信号に対して論理和の演算が行われ、その演算結果が割り込み同期回路65に出力される。ソフトウェアの制御の下に、CPUはどのCPUが割り込みをするかを決定する。I/Oプロセッサ又はI/Oコントローラで発生される外部割り込みはまた、バス35のライン69、70を介し

ルラインは、各I/Oプロセッサから各メモリモジュールへの個々のラインを含む。すなわち、I/Oプロセッサからメモリモジュールへのバス要求、メモリモジュールからI/Oプロセッサへのバスグラント、I/Oプロセッサからメモリモジュールへの割り込み要求ライン、及びメモリモジュールからI/Oプロセッサへのリセットラインである。どのメモリモジュールがプライマリであるかを示すラインは、システムステータスバス32を介して各I/Oプロセッサに接続される。第8図のI/Oプロセッサのコントローラすなわちステートマシン126は、コマンドライン、制御ライン、ステータスライン、ラジアルラインからのデータ、内部データ、及びバス28からのコマンドラインからのデータを受信し、さらに、バス24、25の内容を受信し情報をバスに伝送するために保持するラッチ127、128の動作を含むI/Oプロセッサの内部動作を定義する。

メモリモジュールからI/Oプロセッサへのバス24、25での転送は、別々に肯定応答された

メモリモジュール14、15を介してCPUに号として送信される。同様に、CPUはただプライマリモジュール14又は15からの割り込みに応答する。

#### <I/Oプロセッサ>

第8図において、1個のI/Oプロセッサ26又は27が詳細に示される。I/Oプロセッサは2個の同じポート（I/Oバス24への1個のポート121とI/Oバス25への他のポート122）を備える。各I/Oバス24、25は、32ビット双方向多重アドレス/データバス123（32ビットの外に4ビットのパリティを含む）、リード、ライト、ブロックリード、ブロックライトなどの実行される動作のタイプを定義する双方向コマンドバス124、内部からI/Oプロセッサへの又はバス28のどのロケーションにアドレス指定するかを示すアドレスライン、バ이트マスク、及び最後に、アドレスストローブ、データストローブ、アドレス肯定応答及びデータ肯定応答を含む制御ラインから構成される。バス31のラジアル

アドレスとデータを用いて第9図に示されるプロトコルを使用する。プライマリと指定されたメモリモジュール内のアービトラータ回路110は、I/Oバス24、25の所有権（ownership）のためのアービトレーションを行う。CPUからI/Oへの転送が必要となき、CPU要求がメモリモジュールのアービトレーション論理回路110に出力される。アービトレーション論理回路110がこの要求を承認したとき、メモリモジュールは、アドレスとコマンドを（両バス24、25の）バス123、124に、アドレスストローブが（2つのバス24と25の）第9図の時間T1に主張されたときと同時に、バス125に印加する。コントローラ126がアドレスをラッチ127又は128にラッチさせたとき、アドレス肯定応答がバス125に主張され、次に、メモリモジュールは時間T2にデータを（両バス24、25を介して）バス123に出力し、ライン125にデータストローブを出力する。時間T2の後で、コントローラは、2個のラッチ127、128にデータ

をラッチさせ、データ肯定応答信号がライン125に出力され、そうして、データ肯定応答の受信の際に、両メモリモジュールは、アドレスストロブ信号の主張をやめることによりバス24、25をリリースする。

I/Oプロセッサからメモリモジュールへの転送において、I/OプロセッサがI/Oバスを使う必要があるとき、I/Oプロセッサは、両バス24、25に、ラジアルバス31にラインによってバス要求を主張し、次に、プライマリメモリモジュール14又は15にアービトラーク回路110からバス使用承認信号を持つ。バス使用承認ラインもラジアルラインの1本である。バス使用承認が主張されたとき、コントローラ126は、前の転送が完了されたことを（誤って）意味する、バス125上でアドレスストロブとアドレス肯定応答信号の主張が解除されるまで待機する。そのとき、コントローラ126は、ラッチ127、128からライン123へアドレスを出力させ、コマンドをライン124に出力させ、アドレスス

トロブを両バス24、25のバス125に出力させる。アドレス肯定応答が両バス24、25から受信されたとき、データがアドレス/データバスにデータストロブとともに出力され、転送は、メモリモジュールからI/Oプロセッサへのデータ肯定応答信号で完了される。

ラッチ127と128は、アドレスバス129a、データバス129b、及び制御バス129cを含む内部バス129に接続される。内部バス129は、バス32によって供給されるステイタスを保持するなどのために、コントローラステートマシーン126によって実行されるコマンドをセットアップするために用いられる内部ステイタス・制御レジスタ130をアドレス指定することができる。これらのレジスタ130は、CPUのアドレス空間においてCPUからリードまたはライトのためにアドレス指定可能である。バスインターフェース131は、コントローラ131の制御の下に、VMEバス28と通信する。バス28は、アドレスバス28a、データバス28b、制御バ

ス28c及びラジアル28dを備え、これらの全ラインは、バスインターフェースモジュール29を介してI/Oコントローラ30に接続される。バスインターフェースモジュール29は、一方又は他方の又は両方のI/Oプロセッサからの、1セットだけのバスライン28をコントローラ30に駆動させるためのマルチプレクサ132を備える。コントローラ30の内部で、コマンド、制御、ステイタス、データのレジスタ133があり、(このタイプの周辺コントローラにおいて標準的プラクティスとして)CPU11、12、13から、開始するべきリードとライトのためにアドレス指定可能であり、I/O装置における動作を制御する。

VMEバス28上での各I/Oコントローラ30は、BIM29のマルチプレクサ132を介して両I/Oプロセッサ26、27との接続機能を備え、いずれか1個によって制御されるが、CPUによって実行されるプログラムによって一方または他方に限られる。特定のアドレス(又は組の

アドレス)は、各コントローラ30を示す制御・データ転送レジスタ133のために確定され、これらのアドレスは、オペレーティングシステムによりI/Oページテーブル(通常は、ローカルメモリの核データ区分)に維持される。これらのアドレスは、両方ではなく、どちらかのI/Oプロセッサ#1または#2を介してのみアドレス指定が可能であるように、各コントローラ30を関連づける。すなわち、I/Oプロセッサ27と比較すると、ある異なったアドレスは、I/Oプロセッサ26を介して特定のレジスタ133に到達させるために使用される。バスインターフェース131(及びコントローラ126)は、マルチプレクサ132を切り換えて一方または他方からバス28上のデータを受信する。これは、CPUからI/Oプロセッサのレジスタ130へのライトによってなされる。こうして、デバイスドライバがこのコントローラ30をアクセスするためにコールされたとき、オペレーティングシステムはページテーブルにおけるこのアドレスをそのために使用する

る。プロセッサ40は、ライトバッファ50を介してよりもむしろ、バイパスバッファバス52を用いてこれらのコントローラの制御・データ転送レジスタ133へのI/Oライトによってコントローラ30をアクセスする。従って、これらは、これは、回路100によってポートされ、メモリモジュールを通してバス24又は25へ、従って選択されたバス28への同期化されたライト動作である。プロセッサ40は、このライト動作が完了するまでストールする。第8図のI/Oプロセッサボードは、ある誤り（例えば不適当なコマンド、VMEバス28で応答が受信しないまま期限がすぎたこと、実行されたときのパリティチェック）を検出するように形成され、1個の誤りが検出されると、I/Oプロセッサは、バストラフィックへの応答を止め、すなわち、第9図に関連して上述されたアドレス肯定応答とデータ肯定応答を送信することを中止する。これは、バスインターフェース56によってバスフォールトとして検出され、後述されるように割り込みを生じ、可能ならば自

い。もしいずれかのステータスフィールド（ライン33-1または33-2）においてエラーが検出されたなら、又はもしプライマリでないメモリが時間切れになったならば、割り込みはポスト（post）される。

第2の場合、リード転送において、データエラーがプライマリメモリからステータスライン33に指示されたこと、あるいは、プライマリメモリから応答が受信されなかったことが仮定される。CPUは、他方のメモリから肯定応答を待ち、もし他のメモリからのステータスビットにデータエラーが見いだされないならば、バスインターフェース56の回路が所有権（プライマリのメモリステータス）の変化を起こさせ、従って、データが新しいプライマリから正しくリードされたか否かを確認するために、リトライが設定される。もし良好なステータスが新しいプライマリから受信されたなら、次にストールは前のように終了して、割り込みはシステムを更新するために（1個のメモリを悪いと気づき、異なったメモリをプライマ

リと）リトライされる。

#### <エラーリカバリ>

上記バス21、22、23を介しての転送のためのメモリモジュール14と15による応答を評価するために、CPU11、12、13によって用いられるシーケンスは、次に説明される。このシーケンスは、バスインターフェースユニット56におけるステートマシンによって定義されかつCPUによって実行されるコードにおいて定義される。

第1の場合、リード転送において、データの誤りがプライマリのメモリからのライン33にステータスビットに示されないことと仮定する。ここで、メモリ参照によって始められるストールは、各マイクロプロセッサ40で命令実行を続けることを可能にするために、制御バス55と43を介してレディ信号を主張することにより終了する。しかし、肯定応答がライン112において他の（プライマリでない）メモリモジュールから受信されるまで（または時間切れになるまで）、開始されな

りとし）ポストされる。しかしながら、もしデータエラー又は時間切れが新しいプライマリからリードをするという試みから生じたなら、次に割り込みが制御バス55と43を介してプロセッサ40に主張される。

ライトバッファ50がバイパスされたライト転送において、第1の場合では、どちらのメモリからもステータスビットにエラーが示されない。ストールは終了され命令の実行が許可される。再び、もしエラーがどちらかのステータスフィールドに検出されたならば割り込みがポストされる。

ライトバッファ50がバイパスされたライン転送において、第2の場合では、データエラーがプライマリメモリからステータスに指示されるか、または、応答が、プライマリメモリから受け取れない。各CPUのインターフェースコントローラは、他のメモリモジュールからの肯定応答を待つ。そして、もしデータエラーが他のメモリからのステータスに見いだされないならば、所有権の変化が強制され、割り込みがポストされる。しか

し、もしデータエラー又は時間切れが他方の（新しいプライマリの）メモリモジュールのために起こるならば、次に割り込みがプロセッサ40に対して主張される。

ライトバッファ50がイネーブルされたライト転送において、CPUチップはライト動作によってストールされず、第1の場合は、どちらのメモリモジュールからもステータスにエラーが指示されない。転送は終わられて、他のバス転送が開始される。

ライトバッファ50をイネーブルとしたライト転送において、第2の場合、データエラーは主メモリからのステータスに示されるか、又は、応答が主メモリから受信されない。メカニズムは、他のメモリからの肯定応答を待つ。そして、もし他のメモリからのステータスにデータエラーが見い出されないならば、次に所有権の変化が強行され、割り込みはポストされる。しかし、もしデータエラー又は時間切れが他のメモリにおいて生じるならば、次に割り込みがポストされる。

る。しかし、すべてのライト動作が2個のメモリで実行されるので、この第1ステップが達成される方法は、良好なメモリモジュールのロケーションをリードし、次にこのデータを両メモリモジュール14と15の同じロケーションにライトすることである。これは、通常の動作が進行中に、実行中のタスクに挿入されて行われる。I/Oバス24又は25から生じるライトは、第1ステップでのこの再生ルーチンにおいて無視される。こうしてすべてのロケーションにライトされた後は、次のステップは、I/Oアクセスもまたライトされることを除いて第1ステップと同じである。すなわち、I/Oバス24又は25からのI/Oライトは、実行するタスクにおいて通常のトラフィックにおいて発生するときに、良好なメモリのすべてのロケーションをリードしこの同じデータを両メモリモジュールの同じロケーションにライトすることを挿入して実行される。この第2ステップでモジュールがゼロから最大アドレスまでアドレス指定されたときに、両メモリは同一になる。こ

メモリモジュール14又は15は、いま説明したメカニズムによって一旦決定されると、フォールト条件がオペレータに対し信号として示されるが、システムは動作を続けることができる。オペレータは、おそらく故障のモジュールを含むメモリボードを交換することを希望するだろう。これは、システムが起動され動作している間に行うことができる。次に、システムは、停止せずに新しいメモリボードを再統合できる。このメカニズムは、ソフトのエラーによってライトを実行できないがテストで良好なとされ物理的に交換する必要がないメモリモジュールを再生するためにも役立つ。タスクは、データが他方のメモリモジュールと同じである状態にそのメモリモジュールに戻すことである。この再生モードは、2ステップのプロセスである。まず、メモリがイニシャライズされておらず、パリティエラーを含むかも知れないことを仮定する。そこで良好なパリティの良好なデータが、すべてのロケーションに書き込まなければならない。これは、この点ですべてゼロであ

の第2の再生ステップの間に、CPUとI/Oプロセッサの両方が、エラーなしに全ての動作を行うようにメモリモジュールが再生されることを期待する。I/Oプロセッサ26、27は、データリード転送の間に再生されるメモリモジュールによって示されるデータを使用しない。再生プロセスが完了した後で、再生されたメモリは、（必要ならば）プライマリと指定できる。

同様な再プロセスがCPUモジュールに対しても備えられる。1個のCPUが（メモリポート回路100による場合等のように）故障と検出されるとき、他の2個のCPUは動作を続けるが、悪いCPUボードは、システムを停止せずに交換できる。新しいCPUボードがオンボードROM63から起動自己テストルーチンを実行するとき、他のCPUにこのことを示す信号を出力して、再生ルーチンが実行される。まず、2個の良好なCPUがその状態をグローバルメモリにコピーし、次に全3個のCPUが「ソフトリセット」を実行する。ここで、CPUはROM内のイニシャライ

ブルーチンから実行をリセットし開始する。そうして、全CPUは、命令ストリームの正確に同じ点に来て、同期化され、次に、保存されていた状態が全3個のCPUにコピーして戻され、前に実行されていたタスクが実行される。

上述したように、各メモリモジュール内のポート回路100は、全3個のCPUが同一のメモリ参照をしているか否かを決定する。もしそうならば、メモリ動作は、完了まで進むことを許可される。もしそうでなければ、CPU故障モードに入る。ポート回路100によって検出されるように、異なるメモリ参照を送信するCPUは、バス33-1及び/又は33-2で戻されるステータスで同定される。割り込みはポストされ、ソフトウェアは引き続いて故障CPUをオフラインとする。このオフラインステータスは、ステータスバス32に反映される。故障が検出されているメモリ参照は、3つから2つを選択するポートに基づき完了することを許可される。つぎに、悪いCPUボードが交換されるまで、ポート回路100は、メ

したようにソフトウェアによってI/Oプロセッサに限定されたコントローラ30は、ソフトウェアによって他方のI/Oプロセッサへスイッチされる。オペレーティングシステムは、同じコントローラに対する新しいアドレスを用いてI/Oページテーブルのアドレスを書き直し、その後は、コントローラは他方のI/Oコントローラ26又は27に限定される。エラーすなわち故障は、バスインターフェース56でバスサイクルを終えるバスエラーによって検出でき、例外の原因を決定する例外処理ルーチンを通して域内に急送される例外が発生し、次に、I/Oテーブルのアドレスを書き換えることにより全コントローラ30を、誤ったI/Oプロセッサ26又は27から他方へ動かす。

バスインターフェース56はいま説明したようにバスエラーを検出すると、故障は、再統合スキームが使用される前に分離されねばならない。1個のI/Oプロセッサ26または27へ、あるいは1個のバス28の1個のI/Oコントローラ3

メモリ参照の進行を許可する前に、2個の良好なCPUからの2個の同一のメモリ要求を必要とする。システムは、通常は、1個の(2個でなく)CPUオフラインで動作を続けるように構成されている。しかし、1個の良好なCPUだけで動作することが希望されるならば、別の方法が利用できる。CPUは、もし異なるデータがメモリ要求で検出されるならばポート回路100によって、又は時間切れによって、故障とポートされる。もし2個のCPUが同一のメモリ要求を送信するが、第3のCPUがあらかじめ選択された時間切れ期間にどんな信号も送信しないならば、CPUは故障と仮定され、前のようにオフラインとされる。

システムのI/O装置は故障の場合にソフトウェア再統合のためのメカニズムを備える。すなわち、CPUとメモリモジュールコアは、いま説明したように、故障に対して保護されたハードウェアである。しかし、システムのI/O部分は故障に対して保護されたソフトウェアである。1個のI/Oプロセッサ26または27が誤ると、上述

0へ(すなわち1個のI/O素子における1個の制御レジスタ又はステータスレジスタ、又はデータレジスタへ)のいずれかにCPUがライト動作を行うとき、これは、メモリモジュールにおけるバイパス動作であり、両メモリモジュールは動作を実行して、2個のI/Oバス24と25にそれを通過させる。2個のI/Oプロセッサ26と27は、ともにバス24と25をモニタし、パリティをチェックし、コントローラ126を介して適正なシンタックスでコマンドをチェックする。例えば、もしCPUがI/Oプロセッサ26または27内のレジスタにライトを実行するならば、もしどちらかのメモリモジュールが正当なアドレス、正当なコマンド及び正当なデータを示すならば(パリティエラーがないことと適正なプロトコルによって証明されるように)、アドレス指定されたI/Oプロセッサは、アドレス指定されたロケーションにデータをライトし、ライト動作が成功して完了したという肯定応答指示でメモリモジュールに応答する。両メモリモジュール14と15は、I

／Oプロセッサ26又は27からの応答（第9図のアドレスとデータの肯定応答信号）をモニタして、両メモリモジュールは、ライン33-1と33-2の動作ステータスでCPUに回答する。（もしこれがリードであるならば、プライマリのメモリモジュールのみがデータを戻すが、しかし両方がステータスを戻す。）CPUは、両方がライトを正しく実行したか、1個だけであったか、無しであったかを決定できる。もし1個だけが良好なステータスを戻し、それがプライマリならば、所有権を変える必要は無い。しかしもしバックアップが良好に戻され、プライマリが悪く戻されるならば、所有権の変更が強行され、正しく実行したものをプライマリにする。どちらの場合も、割り込みが故障を報告するために入れられる。この点で、CPUは、悪いのがメモリモジュールであるか、メモリモジュールの下流側の何かであるかを知らない。それで、同様なライトがI/Oプロセッサに対して試みられる。しかし、これが成功するならば、メモリモジュールが悪いことを必ずしも

サがバス24又は25を悪化させているならば、そのバスドライバは、リセットによって切れ、そうして、もしオンラインI/Oプロセッサへの通信のリトライが（両バス24と25を介して）良好なステータスを返すならば、リセットI/Oプロセッサが故障であることが分かる。とにかく、各バスエラーに対して、あるタイプの故障分離シーケンスが実行され、どのシステム部品がオフラインにしなければならないかを決定する。

#### <同期>

図示される実施例において使用されるプロセッサ40は、第4図と第5図を参照して上に説明したように、重なった命令実行を行うパイプラインアーキテクチャである。この実施例において使用される同期技法は、サイクル計数、すなわち、命令が実行される毎に第2図のカウンタ71とカウンタ73をインクリメントすることによるので（米特許出願第118,503号に一般的に開示されているように）、何がプロセッサ40における命令の実行であるかを定義しなければならない。まっ

証明する必要が無い。なぜなら、初めにアドレス指定されたI/Oプロセッサが例えばバス24または25のラインに接続され、パリティエラーを起こしたからである。それで、システムは、選択的に、I/Oプロセッサのシャットオフと操作のリトライを行い、両メモリモジュールが同じI/Oプロセッサにライト動作を正しく実行できるかを見る。もしそうならば、システムは、交換され再統合されるまで悪いI/Oプロセッサをオフラインにして動作を実行できる。しかし、もしリトライが1個のメモリから悪いステータスをなお与えるならば、メモリをオフラインにでき、あるいは、他の要素において故障がメモリにあるか無いかを確定するために故障分離ステップがさらに採られる。これは、全コントローラ30を1個のI/Oプロセッサ26又は27に切り換えてオフのI/Oプロセッサにリセットコマンドを送り、生きている両メモリモジュールでオンラインのI/Oプロセッサとのリトライの通信を出力することを含む。そして、もしリセットI/Oプロセッサ

すぐな定義は、パイプラインが進むことに命令が実行されることである。コントロールバス43の1本のコントロールラインは、パイプラインがストールされることを示す信号ラン#である。ラン#が高レベルであるときは、パイプラインはストールされ、ラン#が低レベル（論理0）であるときは、パイプラインは各マシンサイクル毎に進む。このラン#信号は、数値プロセッサ46においてプロセッサ40のパイプラインをモニタするために用いられ、そうして、このプロセッサ46は、関連するプロセッサ40とともにロックステップでランすることができる。コントロールバス43でのこのラン#信号は、クロック17とともにランサイクルを計数するためにカウンタ71と73によって用いられる。

好ましい実施例において、カウンタレジスタ71のサイズは、4096すなわち $2^{12}$ に選ばれる。これが選択された理由は、クロック17に使用される結晶発振子の許容範囲が、平均して約4Kランサイクルにおけるドリフトがプロセッサチップ

40によってランされるサイクル数において1個のスキューすなわち差を生じるようなものだからである。この差は、以下に説明されるように割り込み同期の正しい動作を正当に許容するようなものである。1つの同期メカニズムは、カウンタ71がオーバーフローするときはいつでもCPUに同期を起こさせるように作用を強いることである。1つのそのような作用は、カウンタ71からのオーバーフロー信号OVFLに対応してキャッシュミスを強いることである。これは、次の1キャッシュのためのコントロールバス43の誤ったミス信号(例えばセットされないタグバリッドビット(TagValid bit))を単に発生することによって行うことができ、こうしてキャッシュミス例外ルーチンをエンターさせ、その結果生じたメモリ参照は、任意のメモリ参照が行われるように同期を確立させる。カウンタ71のオーバーフローに対して同期させる他の方法は、プロセッサ40にストールをさせることである。これは第2図の論理回路71aを介してコントロールバス43にCPビジー

的にはそれ自身のクロックで自由にランしていて、同期イベントが発生する場合を除いて、実質的に相互に結合されていない。マイクロプロセッサが第4図と第5図に示されたように使用される事実は、単独のクロックを用いてのロックステップ同期をより困難にしている、性能を低下させている。また、ライトバッファ50の使用は、プロセッサを結合しないように役立ち、ましてプロセッサの密接な結合に有効ではないであろう。同様に、命令キャッシュとデータキャッシュ及びTLB83を用いた仮想メモリ管理からなる高性能は、密接な結合が使用されたならばさらに困難になり、性能は悪影響を受けるだろう。

割り込み同期技法は、リアルタイムといわゆる「仮想タイム」を区別しなければならない。リアルタイムは、外部の実際の時間、壁の時計の時間であり、秒単位で測定され、又は便宜上例えば60 nsecの分割であるマシンサイクルで測定される。もちろん、クロック発生器17はそれぞれリアルタイムでクロックパルスを発生する。仮想

(コプロセッサビジー) 号を発生するオーバーフロー信号OVFLを用いて行うことができる。このCPビジー信号はCPビジーが主要されなくなるまで、常にプロセッサ40にストールをエンターすることになる。全3個のプロセッサは、同じコードを実行していてそのカウンタ71に同じサイクルを計数するので、このストールをエンターする。しかし、CPUがストールをエンターする実際の時間は変化する。論理回路71aは、入力R#を介して他の2個のプロセッサのバス43からラン#を受信し、そうして全3個がストールしたときに、CPビジー信号がリリースされ、プロセッサは、再び同期してストールから出る。

こうして、2つの同期技法が説明された。第2の技法では、同期はメモリモジュールの回路10におけるメモリ参照から生じ、第2の技法では、いま説明したように、カウンタ71のオーバーフローによって生じる。さらに、以下に説明されるように、割り込みが同期化される。しかし、注意すべき重要なことは、プロセッサ40は、基本

タイムは、各プロセッサチップ40の内部サイクル計数タイムであり、各のサイクルカウンタ71と73で測定される。すなわち、プロセッサチップによって実行される命令の命令数であり、ある任意の開始点からの命令において測定される。第10図を参照して、リアルタイム( $t_0$ から $t_{12}$ )として示される)と仮想タイム(命令数(カウントレジスタ73のモジュール16計数)1<sub>0</sub>から1<sub>12</sub>として示される)との間の関係が図示される。第10図の各行は、1個のCPU-A、-B又は-Cのサイクル計数であり、各列は、リアルタイムでの「点」である。CPUに対するクロックは、位相がずれ易い。そこで、実際の時間の相関は、第10a図に示されるようなものであり、ここで、命令数(列)は完全には並んでおらず、すなわち、サイクル計数は、並べられたリアルタイムマシンサイクルの境界で変化しない。しかし、第10図の図示は、説明の目的には十分である。第10図において、リアルタイム $t_0$ でCPU-Aは第3の命令にあり、CPU-Bは計数9にあり9番目の

命令を実行していて、CPU-Cは4番目の命令にある。リアルタイムも仮想タイムも進むことが可能なのであることに注意する。

CPUのプロセッサチップ40は、リソースが利用出来ないある条件の下でストールする。例えば、ロードまたは命令フェッチの間のDキャッシュ45またはIキャッシュ44のミス、ライトバッファ50がストア動作の間に一杯であるという信号、コプロセッサ46がビジーである(コプロセッサが、データ依存性又は制限された処理リソースにより取り扱えない命令を受信した)というコントロールバス43を介しての「CPビジー」信号、またはマルチプライア/デバイダ79がビジーである(内部のマルチプライア/デバイダ回路が、プロセッサが結果レジスタをアクセスしようとしたときに動作を完了していなかった)ことである。もちろん、キャッシュ44と45は、プロセッサ40による介在なしに状態を変化しない「パッシブリソース」である。しかし、残りのものは、プロセッサがなんら作用しなくても状態を変化出来る

Cは、t<sub>1</sub>でグローバルメモリ14又は15へのアクセスを開始し、グローバルメモリのコントローラ117は、第1のプロセッサCPU-Cがメモリアccessの開始を信号するときにメモリアccessを開始する。コントローラ117は、CPU-BとCPU-Cがそれぞれメモリアccessに必要な8クロックより少なくストールするけれども、アクセス8クロック遅れてt<sub>2</sub>で完了する。その結果、全CPUは、リアルタイムでも仮想タイムでも同期される。この例は、また、DRAM104へのアクセスの重複と回路100でのポーティングとの利点を示す。

インターロックストールは、パッシブリソースストールから異なった状況を示す。1個のCPUは、他のCPUが全くストールをしないときにインターロックストールをすることが出来る。第12図を参照して、ライトバッファ50によって起こされるインターロックストールが図示される。CPU-AとCPU-Bのサイクル計数が示され、CPU-AとCPU-Bのライトバッファ50から

アクティブリソースである。例えば、ライトバッファ50は、(プロセッサが他のストア動作を行わない限り)プロセッサによる作用なしにフルからエンプティに変化する。そこでストールに、パッシブリソースのストールとアクティブリソースのストールの2つのタイプがある。アクティブリソースのストールは、インターロックストールと呼ばれる。

CPU-A、-B、-Cで実行されるコードストリームが同一であるので、3個のCPUのキャッシュ44と45のようなパッシブリソースの状態は、仮想タイムの総ての点で必然的に同じである。もしストールがパッシブリソース(例えばデータキャッシュ45)での衝突の結果であれば、全3個のプロセッサはストールを行い、ただ1つの変数は、ストールの長さである。第11図を参照して、キャッシュミスがt<sub>1</sub>で発生し、このミスの結果のグローバルメモリ14又は15へのアクセスが8クロック(実際には8クロック以上であってもよい)の時間がかかったと仮定する。この場合、CPU-

のフル(full)フラグA<sub>50</sub>とB<sub>50</sub>が、サイクル計数の下に示される(ハイレベルすなわち論理1はフルを意味し、ローレベルすなわち論理0はエンプティを意味する)。CPUはストア動作が実行される毎に、フルフラグの状態をチェックする。もしフルフラグがセットされるならば、CPUは、フルフラグがクリアされストア動作を完了するまでストールする。ライトバッファ50は、もしストア動作がバッファを満たすならば、フルフラグをセットし、ストア動作がバッファから1ワードを流出して次のCPUストア動作のための位置をフリーにするときはいつでもフルフラグをクリアする。時間t<sub>2</sub>で、CPU-Bは、3クロックCPU-Aの先にあり、ライトバッファは共にフルである。ライトバッファがグローバルメモリにライト動作を行っているとは仮定すると、このライトがt<sub>3</sub>の間に完了するとき、ライトバッファフルフラグはクリアされる。このクリアは、リアルタイムでt<sub>3</sub>に同期して起こるが(第11図に図示される理由により)、仮想タイムでは同期していな

い。いま、サイクル計数1.での命令は、ストア動作であると仮定すると、CPU-Aは、ライトバッファフルフラグがクリアされた後で、このストアを実行するが、しかし、CPU-Bは、このストア命令を実行し、そうして、ライトバッファフルフラグがなおセットされていることを見いだして3クロックの間ストールをしなければならない。こうして、CPU-Bはストールをするが、CPU-Aはストールをしない。

1個のCPUはストールをするかも知れず他のCPUはストールをしないかも知れないという性質は、サイクルカウンタ71の解釈に制限を課する。第12図において、割り込みがサイクル計数1.7で(CPU-Bが1.命令からストールをしている間に)複数のCPUに示されたと仮定する。サイクル計数1.7に対するランサイクルは、1.7で両CPUに対して起こる。もしサイクルカウンタだけがCPUに割り込みを示すなら、CPU-Aはサイクル計数1.7で割り込みを見るが、CPU-Bはサイクル計数1.7から生じるストールサイク

ルの間に割り込みを見る。そうして、割り込みを示すこの方法が、この2個のCPUに異なった命令での例外、もし全CPUがストールされるか又はストールされていない場合には起こらないような条件が採用される。

サイクルカウンタの解釈についての別の制限は、サイクル計数の検出と作用の実行との間に遅れがあってはならないことである。再び第12図を参照して、割り込みがサイクル計数1.7でCPUに示されるが、実行の制限のために余分のクロックの遅れがサイクル計数1.7の検出とCPUへの割り込みの提示の間に介在すると仮定する。その結果は、CPU-Aが、この割り込みをサイクル計数1.7で確認するが、CPU-Bは、サイクル計数1.7からのストールの間に割り込みを確認して、2個のCPUに異なった命令で例外を採らせる。再び、リアルタイムで命令パイプラインの状態をモニタすることの重要性が図示される。

#### <割り込み同期>

第1図から第3図までの3個のCPUは、単独

の論理プロセッサとして機能することが要求され、従って、3個のCPUのプログラミングモデルが単独の論理プログラミングのプログラミングモデルであることを保証するためにその内部状態に関してある制限を実行することを要求する。誤りモードや診断モードを除いて、上記3個のCPUの命令ストリームは同一であることが要求される。もし同一でなかったら、第6図のポーティング回路100でのグローバルメモリアクセスのポーティングが困難になるだろう。すなわち、ポートするものは、1個のCPUが故障しているのか、異なったシーケンスの命令を実行しているのか分からない。同期スキームは、もし任意のCPUのコードストリームが、その他のCPUのコードストリームから分岐するならば故障が起こったと仮定するように設計される。割り込み同期は、単独のCPUイメージを維持する1つのメカニズムを提供する。

すべての割り込みは、仮想タイムに同期して起こることが要求され、3個のプロセッサCPU-

A、CPU-BとCPU-Cの命令ストリームが割り込みの結果として分岐しないことを保証する(分岐する命令ストリームには他の原因もある。例えば、1個のプロセッサが他のプロセッサによってリードされたデータと異なったデータをリードすること)。仮想タイムに対して非同期に起こる割り込みがコードストリームを分岐させるシナリオは、いくつかある。例えば、プロセスAが完了する前にコンテキストスイッチをオンにさせるがプロセスAが他のCPUで完了した後でコンテキストスイッチをオンにさせる割り込みは、その後のある点で、1個のCPUがプロセスAの実行を続けるが他方のCPUはプロセスAが既に完了しているためプロセスAを実行出来ないという状況をもたらす。もしこの場合に割り込みが仮想タイムに非同期に起こるならば、例外プログラムカウンタが異なるという事実が問題を起こすであろう。例外プログラムカウンタの値をグローバルメモリに書き込む行為は、ポーターが3個のCPUから異なったデータを検出し、ポートフォールトを生

じるという結果になるだろう。

CPUにおけるあるタイプの例外は、本来仮想タイムに同期している。1つの例は、ブレークポイント命令の実行によって生じるブレークポイント例外である。全CPUの命令ストリームが同一なので、ブレークポイント例外は3個のCPUにおける仮想タイムにて同じ点で生じる。同様に、全てのそのような内部例外は、本来仮想タイムに同期して生じる。例えば、TLB例外は本来同期する内部例外である。TLB例外は仮想ページ数がTLB83のどのエントリにも適合しないために生じる。アドレスを解釈するということが（ブレークポイント例外におけるように正確に）単に命令ストリームの機能なので、解釈は、本来仮想タイムに同期する。TLB例外が仮想タイムに同期することを確実にするために、TLB83の状態は全3個のCPU11、12、13において同一でなければならない。これは、TLB83がソフトウェアだけによって変更できるので、保証される。再び、全CPUが同じ命令ストリームを実行

みは、各CPUの個々の命令ストリーム（仮想タイム）と同期である割り込みを生成することが出来ない。どのような種類の同期も無ければ、もしある外部装置がリアルタイム上の時間に割り込みを駆動し、その割り込みがその時間に全CPUに直接示されるならば、3個のCPUは、異なる命令で例外トラップをとり、3個のCPUのアクセプトされない状態が生じる。これは、リアルタイムに同期であるが仮想タイムに同期しないイベント（割り込みの主張）の例である。

複数の割り込みは、第1図から第3図までのシステムにおいて、割り込みについて分散されたポートを実行し、決定されたサイクル計数でプロセッサに割り込みを示すことにより、仮想タイムに同期する。第13図は、第2図の割り込み同期論理回路65のより詳細なブロック図を示す。各CPUは、モジュール14又は15から生じるライン69又は70からの外部割り込みを捕捉する分配器135を含む。この捕捉はあらかじめ決定されたサイクル計数で、例えばカウンタ71から入力

するので、TLB83の状態は常に仮想タイムに同期して変化される。そうして、一般的経験則として、もし行動がソフトウェアにより実行されるなら、その行動は仮想タイムに同期している。もし行動がサイクルカウンタを用いないハードウェアにより実行されるなら、その行動は一般にリアルタイムに同期である。

外部の例外は、本来仮想タイムに同期していない。I/O装置26、27又は28は、3個のCPU11、12、及び13の仮想タイムについて情報を有しない。従って、I/O装置によって発生される全ての割り込みは、以下に説明するように、CPUに示される前に仮想タイムに同期されなければならない。浮動点例外は、浮動点コプロセッサ46がCPU内でマイクロプロセッサ40に堅く結合されるので、I/O装置割り込みと異なっている。

外部装置は、3個のCPUを1つの論理的プロセッサとして見て、CPU間の同期や同期の欠乏についての情報を有しない。従って、外部割り込

ラインCC-4上で信号が出力される計数-4で起こる。捕捉された割り込みは、CPU間バス18を介して他の2個のCPUへ分配される。これらの分配された割り込みは、未決定割り込みと呼ばれる。各CPU11、12、13から1個の3個の未決定割り込みがある。ポート回路136は、未決定割り込みを捕捉出力、全CPUが外部割り込み要求を受信したかを確認するポートを行う。（サイクルカウンタ71で検出される）あらかじめ決定されたサイクル計数で、この例では入力ラインCC-8により受け取られたサイクル8で、割り込みポート136は、ライン137とバス55と43を介して各マイクロプロセッサ40の割り込みピンに割り込みを示す。割り込みを示すために用いられるサイクル計数があらかじめ決定されているので、全マイクロプロセッサ40は、同じサイクル計数で割り込みを受け取り、こうして、割り込みが仮想タイムに同期されている。

第14図は、仮想タイムに対して割り込みを同期するためのイベントのシーケンスを示す。CP

U-A、CPU-B及びCPU-Cと示された行は、リアルタイムでの1点での各CPUのカウンタ71におけるサイクル計数を示す。IRQ\_A\_PENDING、IRQ\_B\_PENDING及びIRQ\_B\_PENDINGと示された行は、ポート136の入力へCPU間バス18を介して結合される割り込みの状態を示す(1は、未決定ビットがセットされていることを意味する)。IRQ\_A、IRQ\_B、及びIRQ\_Cと示された行は、マイクロプロセッサ40の割り込み入力ピンの状態(ライン137の信号)を示し、ここで1は割り込みが入力ピンに存在することを意味する。第14図で、外部の割り込み(EX\_IRQ)は、 $t_1$ でライン69に主張される。もし割り込み分配器135が割り込みを捕捉し、CPU間バス18にサイクル計数4で分配するならば、IRQ\_C\_PENDINGは時間 $t_1$ で1になり、IRQ\_B\_PENDINGは時間 $t_2$ で1になり、IRQ\_A\_PENDINGは時間 $t_3$ で1になる。もし割り込みポート136がサイクル計数8

を捕捉してポートするならば、CPU-Aの割り込みポート136は、他の2個の割り込み未決定ビットがセットされていないとき、時間 $t_1$ でIRQ\_A\_PENDING信号を捕捉する。CPU-Aの割り込みポート136は全てのCPUが外部割り込みを分配していないことを認識し、捕捉された割り込み未決定ビットを保持レジスタ138に出力されて格納される。CPU-BとCPU-Cの割り込みポート136は単独の割り込み未決定ビットをそれぞれ時間 $t_1$ と $t_2$ に捕捉する。CPU-Aの割り込みポートのように、これらのポートは、全ての割り込み未決定ビットがセットされていないことを認識し、こうして、セットされた単独の割り込み未決定ビットが保持レジスタ138に出力されて格納される。各CPUのサイクルカウンタ71は、サイクル計数7に達するとき、ロールオーバーし、サイクル計数0で計数を開始する。外部割り込みはまだ主張されているので、CPU-BとCPU-Cの割り込み分配器135は、それぞれ時間 $t_1$ と $t_2$ で外部割り込みを捕捉す

で割り込み未決定ビットをポートするならば、IRQ\_Cは時間 $t_1$ で1になり、IRQ\_Bは時間 $t_2$ で1になり、IRQ\_Aは時間 $t_3$ で1になる。その結果、割り込みは、リアルタイムでは異なった点であるが仮想タイムでは同一の点(すなわちサイクル計数8)でCPUに示される。

第15図に、第14図に示されたアルゴリズムを必要とするシナリオを変更して示す。ここではサイクルカウンタ71がモジュール8カウンタにより表されることに注意する。外部割り込み(EX\_IRQ)は時間 $t_1$ で主張される。割り込み分配器135はこの割り込みを捕捉し、サイクル計数4でCPU間バス18に割り込みを分配する。CPU-BとCPU-Cが時間 $t_1$ の前にサイクル計数を実行しているので、その割り込み分配器は外部割り込みを捕捉することができない。しかし、CPU-Aは時間 $t_1$ の前にサイクル計数を実行する。その結果、CPU-Aは時間 $t_1$ で外部割り込みを捕捉して分配する。しかし、もし割り込みポート136がサイクル計数7で割り込み未決定ビッ

る。これらの時間は、サイクル計数が4に等しくなったときに対応する。時間 $t_1$ で、CPU-Cの割り込みポートは、CPU間バス18に割り込み未決定ビットを捕捉する。ポート136は、全C外部割り込みを捕捉して分配することを決定し、プロセッサチップ40に割り込みを示す。時間 $t_1$ と時間 $t_2$ に、CPU-BとCPU-Aの割り込みポート136は、割り込み未決定ビットを捕捉し、割り込みをプロセッサチップ40に示す。その結果、全プロセッサチップが同じ命令で外部割り込み要求を受信したことになり、保持レジスタに保存されていた情報は必要でなくなる。

#### <保持レジスタ>

第15図に関して上述に示された割り込みシナリオにおいて、ポート136は、若干のステート割り込み情報を保存するために保持レジスタ138を使用する。特に、保存されたステートは、全CPUでなくいくつかのCPUが外部割り込みを捕捉し分配したことであった。もしシステムが(第15図の状況のように)どんな故障もし無い場合

は、前の例に示したように、外部割り込みが保持レジスタの使用なしに仮想タイムに同期出来るので、このステート情報は必要でない。アルゴリズムは、割り込みポート136が割り込み未決定ビットをあらかじめ決定されたサイクル計数で捕えポートすることである。全ての割り込み未決定ビットが主張されるとき、割り込みは、そのあらかじめ決定されたサイクル計数でプロセッサチップに示される。第15図の例において、割り込みはサイクル計数7でポートされた。

第15図を参照して、もしCPU-Cが誤りをし、誤りモードが割り込み分配器135が正しく機能しないようなものであれば、このとき、もしプロセッサチップ40に割り込みを示す前に全割り込み未決定ビットがセットされるまで割り込みポート136が待つならば、その結果、割り込みは示されるようになることは無い。こうして、ただ1個のCPUのただ1個の誤りが全CPUについての全体の割り込みのチェーンを機能できないようにする。

割り込み未決定ビットがセットされていることを割り込みポートが発見するならば、エラーが1個以上のCPUに存在するはずである。これは、各CPUの保持レジスタ138が割り込みサービス時にクリアされることを仮定する。そのため、保持レジスタの状態は割り込み未決定ビットについての新鮮でない状態を変えない。エラーの場合、割り込みポート136は、プロセッサチップ40に割り込みを示すことができ、同時に、エラーが割り込み同期論理回路によって検出されたことを示す。

割り込みポート136は、実際にはどんなポーティングもせず、その代わり割り込み未決定ビットと保持レジスタ137の状態を検査して、プロセッサチップ40に割り込みを示すか否かと割り込み同期論理回路にエラーを示すか否かを決定するだけである。

#### <モジュールサイクルカウンタ>

第15図の割り込み同期の例は、割り込みカウンタをモジュールNカウンタ（例えばモジュール

保持レジスタ138は、最後の割り込みポートサイクルが全部ではないが少なくとも1個の割り込み未決定ビットを捕獲したことをポート136が知るメカニズムを提供する。割り込みポートサイクルは、割り込みポートが割り込み未決定ビットを捕獲しポートするサイクル計数で起こる。数個の割り込み未決定ビットがセットされる結果となる2つだけのシナリオがある。1つは、第15図に関連して示された示されたシナリオであって、ここでは、外部割り込みは、あるCPUの割り込み分配サイクルの前であるがその他のCPUの割り込み分配サイクルの後に主張される。第2のシナリオでは、少なくとも1個のCPUが、割り込み分配器をディスエーブルにするような誤りをする。もし数個の割り込み未決定ビットだけが割り込みポートサイクルでセットされる理由が第1のシナリオであるならば、割り込みポートは、全割り込み未決定ビットが次の割り込みポートサイクルでセットされることが保証される。従って、もし保持レジスタがセットされていて全部でない割

り込み未決定ビットがセットされていることを割り込みポートが発見するならば、エラーが1個以上のCPUに存在するはずである。これは、各CPUの保持レジスタ138が割り込みサービス時にクリアされることを仮定する。そのため、保持レジスタの状態は割り込み未決定ビットについての新鮮でない状態を変えない。エラーの場合、割り込みポート136は、プロセッサチップ40に割り込みを示すことができ、同時に、エラーが割り込み同期論理回路によって検出されたことを示す。

割り込みポート136は、実際にはどんなポーティングもせず、その代わり割り込み未決定ビットと保持レジスタ137の状態を検査して、プロセッサチップ40に割り込みを示すか否かと割り込み同期論理回路にエラーを示すか否かを決定するだけである。

<モジュールサイクルカウンタ>

第15図の割り込み同期の例は、割り込みカウンタをモジュールNカウンタ（例えばモジュール8カウンタ）として表した。モジュールNサイクルカウンタの使用は、割り込みポートサイクルの概念を可能にすることにより、割り込みポーティングアルゴリズムの説明を簡単にした。モジュールNサイクルカウンタを使用すると、割り込みポートサイクルは、0とN-1（Nはサイクルカウンタのモジュールである）の間にある単独のサイクル計数として説明できる。サイクルカウンタのどんな数も割り込みポートサイクルのために選択でき、サイクル計数は、Nサイクル計数毎に起こることが保証される。モジュール8カウンタに対して第15図に示されるように、割り込みポートサイクルは8計数毎に起こる。割り込みポートサイクルは、モジュールNサイクルカウンタの周期的性質を説明するためにだけここで用いられる。モジュールNサイクルカウンタの特定のサイクル計数にキーとなるどのイベントもNサイクル計数毎に起こることが保証される。明らかに、不定数（すなわち非反復性カウンタ71）は使用できない。

Nの値は、システムに正の効果を持つシステムパラメータを最大にし、システムに負の効果を持つシステムパラメータを最小にするように選択される。まず、いくつかのパラメータが示される。 $C_+$ と $C_-$ は、それぞれ、割り込みポートサイクルと割り込み分配サイクルである(第13図の回路では、これらはそれぞれCC-8とCC-4である)。 $CC-8$ と $CC-4$ の値は、0と $N-1$ ( $N$ はサイクルカウンタのモジュールである)の間の範囲にあらねばならない。 $D_{...}$ は、同期論理回路によって許容され得る3個のプロセッサCPU-A、CPU-B及びCPU-Cの間のサイクル計数ドリフトの最大量である。このプロセッサドリフトは、リアルタイムの1点で各CPUからサイクルカウンタ71のスナップショットをとることにより決定される。ドリフトは、最も遅いCPUのサイクル計数を最速のCPUのサイクル計数から差し引くこと(モジュールN減算としてなされる)により計算される。 $D_{...}$ の値は、 $N$ と $C_+$ と $C_-$ の関数として表される。

時間がクロックサイクル(ランサイクル)のインクリメントで量子化されているので、 $e$ も量子化出来る。こうして、次の式が得られる。

$$\text{方程式(2)} \quad C_+ - C_- < D_{...} - 1$$

ここに、 $D_{...}$ は、サイクル計数の整数値として表される。

次に、最大のドリフトが $N$ の関数として表すことができる。第17図は、 $N=4$ でプロセッサドリフト $D=3$ の場合のシナリオを示す。 $C_+=0$ と仮定する。各プロセッサのサイクル計数0における減算は、命令サイクル計数の商の部分( $Q$ )を表す。サイクル計数がいまモジュールNにて示されるので、サイクルカウンタの値は、 $1/N$ ( $1$ は、時間 $t_1$ 以来実行された命令数である)の剰余である。命令サイクル計数の $Q$ は、 $1/N$ の整数部分である。もし外部割り込みが時間 $t_1$ に主張されるならば、CPU-Aは、時間 $t_1$ に割り込みを捕え分配し、CPU-Bは、時間 $t_1$ に割り込み分配サイクルを実行する。CPU-Aに対する割り込み分配サイクルが $Q=1$ でありCPU-B

まず、 $D_{...}$ は、差 $C_+ - C_-$ の関数として表される。ここに、差演算はモジュールN減算として実行される。これは、 $D_{...}$ を最大にする $C_+$ と $C_-$ の値を選択することを可能にする。第16図のシナリオを参照し、 $C_+=8$ と $C_-=9$ を仮定する。第16図から、プロセッサドリフトは $D_{...}=4$ であると計算出来る。ライン69の外部割り込みは、時間 $t_1$ で主張される。この場合、CPU-Bは、時間 $t_1$ で割り込みを捕え分配する。このシナリオは、前に示された割り込み同期アルゴリズムとつじつまが合わない。なぜなら、CPU-Aが割り込み分配サイクルを行った前にCPU-Bがその割り込みポートサイクルを実行するからである。このシナリオの欠陥は、 $C_+$ と $C_-$ の差よりも更に離れてドリフトすることである。この関係は、形式的に次のように書くことができる。

$$\text{方程式(1)} \quad C_+ - C_- < D_{...} - e$$

ここに、 $e$ は、CPU間バス18に伝達される割り込み未決定ビットのために必要な時間である。前の例では、 $e$ は0と仮定されていた。整時計の

に対する割り込み分配サイクルが $Q=2$ であるので、これは問題を示す。同期論理回路は、問題が無いかのように続行し、こうして等しいサイクル計数でプロセッサに割り込みを示す。しかし、各プロセッサの $Q$ は異なっているので、割り込みは異なった命令で複数のプロセッサに示される。従って、 $N$ の関数としての $D_{...}$ の関係は次式で表される。

$$\text{方程式(3)} \quad N/2 > D_{...}$$

ここに、 $N$ は偶数であり、 $D_{...}$ はサイクル計数の整数として表される。ここで、方程式(2)と(3)は共に標準化理論におけるナイキストの定理に等価であることを示すことができる。方程式(2)と(3)とを結合することによって次式を得る。

$$\text{方程式(4)} \quad C_+ - C_- < N/2 - 1$$

ここに、 $N$ の与えられた値に対して $C_+$ と $C_-$ の最適の値が選択できる。

上述の全方程式は、 $N$ が出来るだけ大きくあるべきであることを示唆する。 $N$ を小さくさせよう

とする唯一の因子は、割り込みの潜在である。割り込みの潜在は、ライン69での外部割り込みの主要とライン137でのマイクロプロセッサチップへの割り込みの提示との間の時間間隔である。どのプロセッサが割り込みの潜在を決定するために使用されるべきかは明快な選択でない。3個のマイクロプロセッサは、クロック源における結晶発振子におけるわずかな違いや他の因子のために異なった速度で動作する。最も高速のプロセッサと、最も遅いプロセッサと、その他のプロセッサがある。システムの性能は最も遅いプロセッサの性能によって最終的に決定されるので、最も遅いプロセッサに関して割り込みの潜在を定義することは合理的である。最大の割り込みの潜在は、

$$\text{方程式(5)} \quad L_{\dots} = 2N - 1$$

であり、ここに、 $L_{\dots}$ は、サイクル計数で表された最大の割り込みの潜在である。最大の割り込みの潜在は、最速のプロセッサの割り込み分配サイクル $C_1$ の後であるが最も遅いプロセッサの割り込み分配サイクル $C_3$ の前に外部割り込みが主

$L_{\dots} = 16/2 + (8 - 4) = 12$  サイクルすなわち0.7ミリ秒である。

#### <ローカルメモリのためのリフレッシュ制御>

リフレッシュカウンタ72は、カウンタ71と71aがまさに計数するのと同様に、(マシンサイクルでなく)非ストールサイクルを計数する。目的は、リアルタイムよりはむしろ仮想タイムで測定して、同じサイクル計数で各CPUにリフレッシュサイクルを導入することである。好ましくは、各CPUは、命令ストリームにおいて他のCPUと同じ点でリフレッシュサイクルを課する。ローカルメモリ16のDRAMは、グローバルなメモリについて上述したように8msec毎に512サイクルの周期でリフレッシュされねばならない。こうして、カウンタ72は、512の1行をアドレスして、15msec毎に1回DRAM16にリフレッシュコマンドを出力しなければならない。もしメモリ動作がリフレッシュの間に要求されたならば、リフレッシュが終了するまでビジー応答が生じる。しかし、各CPUにそれ自身のローカ

要求されたときに、最大の割り込みの潜在が起こる。平均の割り込みの潜在の計算は、最速のプロセッサの割り込み分配サイクルの後でかつ最も遅いプロセッサの割り込み分配サイクルの前に外部割り込みが起こる確率に依存するので、さらに複雑である。この確率は、多数の外部因子によって順番に決定されるプロセッサ間のドリフトに依存する。もしこれらの確率が0であるならば、平均の潜在は次の式で表される。

$$\text{方程式(6)} \quad L_{\dots} = N/2 - (C_1 - C_3)$$

これらの関係式を用いて、 $N$ 、 $C_1$ 、及び $C_3$ の値が、 $D_{\dots}$ と割り込みの潜在とに対するシステムの要請を使用して決定される。例えば、 $N = 128$ 、 $(C_1 - C_3) = 10$ 、 $L_{\dots} = 74$ 又は約4.4マイクロ秒(ストールサイクルなしで)を選択する。4ビット(4つの2進ステージ)71aが割り込み同期カウンタとして使用され、分配出力とポート出力が説明したようにCC-4とCC-8にある好ましい実施例を用いて、 $N = 16$ 、 $C_1 = 8$ 、 $C_3 = 4$ であることが分かり、そうして、

ルメモリのリフレッシュをリアルタイムで他のCPUに独立に処理させることは、CPUを同期から外れさせ、従って、余分な制御が必要になる。例えば、もし丁度除算命令が始まるようにリフレッシュモードがエンターされるならば、タイミングは、1個のCPUが他のCPUより2クロックだけ長くかかるようなタイミングになる。又は、もし割り込み可能でないシーケンスがより高速なCPUによりエンターされ他のCPUがこのルーチンにエンターする前にリフレッシュに入るならば、CPUは、相互に離れていく。しかし、これらの問題のいくつかを避けるためのサイクルカウンタ71を(リアルタイムの代わりに)使用することは、ストールサイクルが計数されないことを意味する。そして、もしループに入って多くのストールを生じさせるならば(7対1のストール・ラン比を生じさせることが可能ならば)、周期が15msecの数値から著しく減少されないならば、リフレッシュの仕様に合わず、性能を劣化させる。この理由のために、第2図に示されるように、ス

トールサイクルは第2カウンタ72aでも計数され、このカウンタがリフレッシュカウンタ72で計数されるのと同じ数に達する毎に、追加のリフレッシュサイクルが導入される。例えば、リフレッシュカウンタ72は、カウンタ71と歩調を合わせて、2<sup>8</sup>すなわち256ランサイクルを計数し、オーバーフローのときにリフレッシュ信号が制御バス43を介して出力される。一方、カウンタ72aは、(ラン#信号とクロック17に応答して)2<sup>8</sup>ストールサイクルを計数し、オーバーフローする毎に第2カウンタ72aがインクリメントされる(カウンタ72bは単に8ビットカウンタ72aのためのビット9から11であってもよい)。そうして、リフレッシュモードが最後にエンターされ、CPUはカウンタレジスタ72bの数によって示される多数の追加のリフレッシュを行う。こうして、もし長期間のストールインテンシブな実行が起こるならば、リフレッシュの平均数は、15マイクロ秒毎に1つの範囲内にあり、もし7×256までのストールサイクルが介在されるなら

かし、8Mバイト以下のアドレスは、CPUモジュールそれぞれ自身内でローカルメモリ16にアクセスする。性能は、ローカルメモリ16で実行されるアプリケーションにより使用されるメモリをより多く配置することにより改善される。そして、もしメモリチップが高密度でより低コストでより高速で利用できるならば、追加のローカルメモリが、追加のグローバルメモリと同様に付加される。例えば、ローカルメモリが32Mバイトであって、グローバルメモリが128Mバイトであってもよい。一方、非常に低コストのシステムが必要ならば、性能は主要な決定的なファクタではなく、システムは、ローカルメモリなしに動作でき、そのような構成では性能の不利益が高いけれども、すべてのメインメモリはグローバルメモリエリア(メモリモジュール14と15)である。

第18図のマップのローカルメモリ部分141の内容は、3個のCPU11、12及び13における内容と同一である。同様に、2個のメモリモジュール14と15は、どの与えられた瞬間でも

ば、最後にリフレッシュモードに行くときにリフレッシュされた行の数が名目上のリフレッシュ速度まで追い付くので、リフレッシュサイクルを任意に短くすることにより性能の劣化はない。

#### <メモリ管理>

第1図から第3図までのCPU11、12、及び13は、第18図に図示されるように組織されたメモリ空間を備える。ローカルメモリ16が8Mバイトであり、グローバルメモリ14又は15が32Mバイトである例を用いて、ローカルメモリ16が、キャッシュすなわち別のメモリ空間であるよりはむしろ、CPUメモリアクセス空間の同じ連続的な0から40Mバイトまでのマップの一部である。0から8Mバイトまでの部分を(3個のCPUモジュールで)3重化し、8から40Mバイト部分を2重化しているが、論理的には単に1つの0から40Mバイトまでの物理アドレス空間があるだけである。バス54で8Mバイトを超えたアドレスは、バスインターフェース56にメモリモジュール14と15に要求をさせるが、し

その空間142内の同じデータを全く同様に含む。ローカルメモリ部分141内にはUNIXオペレーティングシステムのための核143(コード)が格納され、このエリアは、各CPUのローカルメモリ16の固定された部分内に物理的にマッピングされる。同様に、核データは、各ローカルメモリ16の固定されたエリア141に割り当てられる。ブートアップの時を除いて、これらのブロックは、グローバルメモリ又はディスクへ、又はグローバルメモリ又はディスクから交換されない。ローカルメモリの他の部分145は、ユーザプログラム(及びデータ)のページのために使用され、これらのページは、オペレーティングシステムの制御の下にグローバルメモリ14と15のエリア146に交換される。グローバルメモリエリア142は、エリア146におけるユーザーページのためのステージングエリア(staging area)として、またエリア147におけるディスクバッファとして使用される。もし全CPUが1ブロックのデータのライトを行うコード又はローカルメモリ

16からディスク148へのコードを実行するならば、ディスクバッファエリア147にコピーをするための時間はI/Oプロセッサ26と27に直接にそしてI/Oコントローラ30を介してディスク148にコピーをする時間に比べて無視できるので、シーケンスは、その代わりディスクバッファエリア147にライトを行うことである。次に、全CPUが他のコードの実行を進める間に、このディスクにライトをする動作が行われて、全CPUに対してトランスペアレントに、そのブロックをエリア147からディスク148へ移動する。同様な方法で、グローバルメモリエリア146は、ディスク以外のI/Oアクセス（例えばビデオ）の同様な処理のために、I/Oステージングエリア149を含んでマッピングされる。

第18図の物理的メモリマップは、各CPU内のプロセッサ40の仮想メモリ管理システムと関連する。第19図は、実施例において使用されたR2000プロセッサチップの仮想アドレスマップを図示する。しかしながら、ページングと保護

2Gバイトの“kuseg”として参照される単独の様な仮想アドレス空間150を利用できる。各仮想アドレスはまた、最大64個のユーザープロセスのための一義的仮想アドレスを形成するために、6ビットのプロセスアイデンティファイア(PID)フィールドを用いて拡張される。ユーザーモードにおけるこのセグメント150までのすべての参照は、TLB83を介してマッピングされ、キャッシュ144と145の使用は、TLBエントリにおける各ページエントリのためのビットセッティングによって決定される。すなわち、あるページは、キャッシュ可能で有り得るし、あるページはプログラマによって特定されるのでキャッシュ可能でない。

核モードにあるとき、仮想メモリ空間は、第19図の両エリア150と151を含む。この空間は、4つの別々のセグメントkusegエリア150、ksegoエリア152、kseglエリア153及びkse g2エリア154を有する。核モードのためのkusegエリア150のセグ

メカニズムを備えた仮想メカニズム管理を支持する他のプロセッサチップが対応する特徴を備えるであろうことが理解される。

第19図において、2つの別々の2Gバイトの仮想アドレス空間150と151が図示される。プロセッサ40は、2つのモード、ユーザーモードと核モード、の1つで動作する。当該プロセッサはただ、ユーザーモードにおいてエリア150をアクセスでき、もしくは核モードにおいて両エリア150と151をアクセスすることができる。核モードは、多くの計算機に備えられている監視モードと同様である。プロセッサ40は、例外が検出されてモードを核モードに強いるまでは、通常はユーザーモードで動作するように構成され、ここで、例外からのリストア(RFE)命令が実行されるまで核モードにとどまる。メモリアドレスが翻訳されるかわりマッピングされる方法は、マイクロプロセッサのオペレーティングモードに依存し、これはステイクスレジスタの1ビットによって定義される。ユーザーモードにあるときに、

メントは、ユーザーモードの“kuseg”エリアに対応して2Gバイトのサイズを有する。従って、核モードにおいて、プロセッサはまさにユーザーモードの参照におけるようにこのセグメントに対して参照を行って、ユーザーデータへの核アクセスを効率化する。kusegエリア150は、ユーザーコードとユーザーデータを保持するために使用される。しかし、オペレーティングシステムは、しばしばこの同じコード又はデータを参照することを必要とする。上記ksegoエリア152は、物理的アドレス空間の初めの512Mバイトに直接にマッピングされる512Mバイトの核物理的アドレス空間であり、キャッシュされるが、TLB83を使用しない。このセグメントは、核実行可能コードとある核データのために使用され、ローカルメモリ16内に第18図のエリア143によって表される。上記kseglエリア153は、ksegoエリアと同様に、物理的アドレス空間の初めの512Mバイトに直接にマッピングされ、キャッシュされず、TLBエントリを用いな

い。kseg1エリアは、キャッシュされないことだけがkseg0エリアと異なる。kseg1エリアは、I/Oレジスタ、ROMコード及びディスクバッファのためのオペレーティングシステムによって使用され、第18図の物理的マップのエリア147と149に対応する。kseg2エリア154は、1Gバイトの空間であり、kseg0エリアのように、キャッシュを用い又は用いずに、任意の物理的アドレスに仮想アドレスをマッピングするためのTLB83エントリを使用する。このkseg2エリアは、ユーザーモードにおいてアクセスできず、核モードにおいてのみアクセスできるということだけが、kseg0エリア150と異なる。オペレーティングシステムは、ユーザーページテーブル(メモリマップ)のためと動的に割り当てられるデータエリアのために、コンテキストスイッチに再びマッピングしなければならないスタックとパープロセスデータ(per-process data)のためにkseg2エリアを使用する。kseg2エリアは、全てか無かのア

ドレス38-43と比較される。もし対の一方が64の64ビットTLBエントリのいずれかに見いだされるならば、対となったエントリのビット12-31でのページフレーム数PFNは、(他の基準が適合することを仮定して)第3図のバス82と42を介した出力として使用される。TLBエントリにおける他の1ビットの値は、N、D、V及びGを含む。ここで、Nはキャッシュできない指標であり、もしセットされれば、ページはキャッシュできず、プロセッサは、キャッシュ44又は45をまずアクセスする代わりにローカルメモリ又はグローバルメモリをアクセスする。Dは、ライトプロテクトビットであり、もしセットされれば、ロケーションが「よこれ」でいて、従って、ライト可能であるが、もし0ならば、ライト動作はトラップを起こすことを意味する。Vビットは、セットされれば、正当であることを意味し、単に正当なビットを再セットするだけでTLBエントリをクリアできることを意味する。このVビットは、このシステムのページのスワッピング配位におい

て、ページが必要とするよりはむしろ、パーページベース(per page basis)への選択的キャッシングとマッピングを可能にする。

マイクロプロセッサチップのレジスタ76又はPC80とバス84での出力に発生される32ビットの仮想アドレスは、第20図に示される。ここで分かるように、ビット0-11は、第3図のバス42でのアドレスの下位12ビットとして無条件に使用されるオフセットであり、ビット12-31は、ビット29-31がkseg0エリア、kseg1エリア及びkseg2エリアの間で選択する仮想ページ数(VPN)である。現在実行中のプロセスのためのプロセスアイデンティファイア(PID)は、TLBによってもアクセス可能なレジスタ内に格納される。64ビットのTLBエントリは、同様に第20図に表され、ここで分かるように、仮想アドレスからの29ビットVPNは、64ビットエントリのビット44-63に位置される20ビットVPNフィールドと比較され、一方、同時に、PIDはビッ

て、ページがローカルメモリにあるかグローバルメモリにあるかを示すために使用される。Gビットは、正当なTLB翻訳のためのPIDマッチの要請を無視するグローバルアクセスを許可するためにある。

装置コントローラ30は、ローカルメモリに対してDMAを直接に行うことができない。従って、グローバルメモリは、DMAタイプのブロック転送(典型的にはディスク148などから)のためのステージングエリアとして使用される。CPUは、コントローラ(すなわちプログラムされたI/Oによって動作を開始した制御するために、コントローラ30において直接に動作を実行することができる。しかしながら、コントローラ30は、グローバルメモリに対するDMAを除いて、DMAを行うことができない。コントローラ30は、VMEバス(バス28)マスクになることができ、I/Oプロセッサ26又は27を介してメモリモジュール14と15内のグローバルメモリに直接にリード動作とライト動作を行う。

グローバルメモリとローカルメモリ（及びディスク）との間のページのスワッピングは、ページフォールトとエージングプロセスとの一方によって開始される。プロセスが実行中でありグローバルメモリ又はディスクにあるページから実行すること又はそのページからアクセスを試みることに、ページフォールトが生じる。すなわち、TLB 83は、ミスを示し、トラップが生じるであろう。従って、核のローレベルトラップコードがページのロケーションを示し、ページのスワッピングを開始するためのルーチンがエンターされる。もし必要とされるページがグローバルメモリ内にあるならば、一連のコマンドがDMAコントローラに送られて、最も少なく最近使用されたページをローカルメモリからグローバルメモリに書き込み、その必要とされたページをグローバルメモリからローカルメモリに読み出す。もしそのページがディスクにあるならば、コマンドとアドレス（セクタ）が、ディスクに行ってそのページを得るためにCPUからコントローラ30に書

ないページについてマークしながら周期的にローカルメモリ内のページを通過していく。タスクスイッチはそれ自身ページのスワッピングを開始しないが、その代わり、新しいページがページフォールトをつくり始めたとき、ページは必要なだけスワッピングされ、スワッピングのための候補は、最近は使用されていないものである。

もしメモリ参照がなされTLBミスが示されるが、しかしTLBミス例外から生じるページテーブルルックアップがそのページがローカルメモリ内にあることを示すならば、このページがローカルメモリ内にあることを示すためにTLBエントリがなされる。すなわち、プロセスは、TLBミスが起こったときに例外をとり、（核データ区分内の）ページテーブルに行き、テーブルエントリを見付け、TLBに対して書き込み、次に進むことが許される。しかし、もしメモリ参照がTLBミスを示し、ページテーブルが、対応する物理アドレスが（8Mバイトの物理アドレスを越えて）グローバルメモリ内にあることを示すならば、TL

き込まれる。そして、メモリ参照をするプロセスが一時停止される。ディスクコントローラがデータを見付けそれを送信する用意ができたとき、メモリモジュールによって（CPUに到達せずに）使用される割り込み信号が出力されて、グローバルメモリにそのページをき込むためにグローバルメモリへのDMAをディスクコントローラが始めることを許可する。終了したときは、CPUは割り込みされて、DMAコントローラの制御の下にブロック転送を開始して、最も少なく使用されたページをローカルメモリからグローバルメモリへスワッピングし、必要なページをローカルメモリへ読み込む。次に、元のプロセスが再び実行（ラン）可能にされ、その状態は元に戻され、元のメモリ参照が再び生じ、ローカルメモリ内にその必要なページを見付ける。ページのスワッピングを開始するもう一つのメカニズムは、エージングルーチンであり、これにより、オペレーティングシステムは、各ページが最近使用されたか否かについて又グローバルメモリへの押し出しを被ってい

Bエントリがこのページのために実行され、そして、プロセスが再び続くとき、プロセスは、前と同様にTLB内にページエントリを見いだす。さらに一つの例外は、正当なビットが0であって、そのページが物理的にローカルメモリ内にあることを示すために採られる。そして、このときは、例外は、グローバルメモリからローカルメモリにページをスワッピングするルーチンをロードし、そして実行が進むことができる。第3の状況では、もしページテーブルが、メモリ参照のためのアドレスがローカルメモリやグローバルメモリ内に無くディスクにあることを示すならば、システムは、上に示されたように動作し、すなわち、プロセスはランキュー（run queue）を去り、スリープキュー（sleep queue）に入り、ディスク要求がなされ、ディスクがそのページをグローバルメモリに転送しコマンド完了割り込み信号を出力したとき、ページがグローバルメモリからローカルメモリへスワッピングされ、TLBは更新され、次にプロセスは再び実行できる。

## ＜プライベートメモリ＞

メモリモジュール14と15は同じ位置に同じデータを格納でき、全3個のCPU11、12及び13はこれらのメモリモジュールに対して等しいアクセスを行うが、各メモリモジュールにはプライベートメモリとしてソフトウェア制御のもとで割り当てられた小さなエリアがある。例えば、第21図に図示されるように、メモリモジュール位置のマップのエリア155は、プライベートメモリエリアとして呼ばれ、全CPUが「プライベートメモリライト」コマンドをバス59に出力したときのみライト可能である。実施例では、プライベートメモリエリア155は、各CPUモジュールのバスインターフェース56のレジスタ156に含まれるアドレスで出発する4Kのページである。この出発アドレスは、CPUによってこのレジスタ156に書き込むことによってソフトウェア制御のもとで変更できる。プライベートメモリエリア155は、さらに3個のCPUの間で分割される。CPU-Aだけがエリア155aに書

レスの2ビットによって決定される。このプライベートライトの間に、全3個のCPUは、バス57に同じアドレスを示すが、バス58に異なったデータを示す（この異なったデータは、例えばCPUへのステートキューである）。メモリモジュールは、アドレスとコマンドをポートし、アドレスバスに見られたアドレスフィールドの部分によって基づいてただ1個のCPUからデータを選択する。CPUがデータをポートすることを可能にするため、全3個のCPUは、両メモリモジュール14と15内へ、CPUに一義的なステート情報の3個のプライベートライト動作（バス21、22、23に3個のライト動作がある）を行う。各ライト動作の間に、各CPUは、一義的データを送信するが、ただ1個だけが各時間にアクセプトされる。それで、全3個のCPUによって実行されるソフトウェアシーケンスは、(1) ロケーション155aにストア、(2) ロケーション155bにストア、(3) ロケーション155cにストアである。しかしながら、ただ1個のCPUから

き込むことができ、CPU-Bだけがエリア155bに書き込むことができ、CPU-Cだけがエリア155cに書き込むことができる。バス57の1つのコマンド号は、動作がプライベートライトであることをメモリモジュール14と15に知らせるために、バスインターフェース56によってセットされる。そして、これは、ストア命令からプロセッサ40によって発生されたアドレスに対応してセットされる。アドレスのビット（およびライトコマンド）は、（バスアドレスをレジスタ156の内容に比較する）バスインターフェース内のデコーダ157によって検出され、バス57に対する「プライベートメモリライト」コマンドを発生するために使用される。メモリモジュールでは、ライトコマンドがレジスタ94、95及び96で検出され、アドレスとコマンドが全てポート回路100によって良好（すなわち一致している）とポートされたとき、制御回路100は、ただ1個のCPUからのデータをバス101へと通すことを許可し、これは、全CPUからのアド

のデータが実際には各時間に書き込まれ、そのデータはポートされない。なぜならば、異なっており又は異なる可能性があり、そしてポートされるならばフォールトを示す可能性があるからである。次に、全CPUは、全3個のロケーション155a、155b、155cを読んで、ソフトウェアによりこのデータを比較することによってデータポートすることができる。このタイプの動作は、例えば診断に又は原因レジスタ（cause register）データをポートするための割り込みにおいて使用される。

プライベートライトのメカニズムは、フォールト検出と回復において使用される。例えば、もし全CPUがメモリーリード要求をするときにバスエラー（メモリモジュール14又は15がバッドステータス（bad status）をライン33-1または33-2に戻すようなとき）を検出するような場合である。この点で、CPUは、他のCPUがメモリモジュールから同じステータスを受け取っているか否かを知らない。CPUが故障で有り得る

し、そのステイタス検出回路が故障で有り得るし、あるいは、示されたように、メモリが故障で有り得る。それで、故障を分離するために、上述のバスフォールトルーチンがエンターされたときに、全3個のCPUは、前のリードの試みでメモリモジュールからまさに受信したステイタス情報のプライベートライト動作を行う。次に、全3個のCPUは、他のCPUが書き込んだものを読み出し、自分自身のメモリステイタス情報と比較する。もしそれらが一致するならば、メモリモジュールは、オフラインでポートされる。もし一致せず、1個のCPUがメモリモジュールに対して悪いステイタスを示し他のCPUが良好なステイタスを示すならば、CPUはオフラインでポートされる。

#### <フォールトトレラント電源>

第22図を参照して、好ましい実施例のシステムは、上述のCPUモジュール、メモリモジュール、I/Oプロセッサモジュール、I/Oコントローラ、及びディスクモジュールのオンラインでの交換と同様に、故障した電源モジュールをオン

4つの別々の電力分配バスがこれらのバス166と167に含まれる。バルク電源164は、電力バス166-1と167-1を駆動し、バルク電源165は、電力バス166-2と167-2を駆動する。バッテリーパック163は、バス166-3、167-3を駆動し、バス166-1と167-2から再チャージされる。3個のCPU11、12、13は、これらの4個の分配バスの異なる組み合わせから駆動される。

これらの36Vバス166と167に結合された多数のDC-DCコンバータ168が、CPUモジュール11、12及び13、メモリモジュール26と27、及びI/Oコントローラ30を個々に電力を供給するために使用される。バルク電源16と165は、また、3個のシステムファン169と、バッテリーパック162と163のためのバッテリーチャージャに電力を供給する。各システム部品に対するこれらの別々のDC-DCコンバータを備えることにより、1個のコンバータの故障はシステムシャットダウンを生じず、その代

ラインで交換できるフォールトトレラントな電源を使用できる。第22図の回路で、交流電力ライン160は、電力分配ユニット161に直接に接続され、このユニット161は、電力ラインのろ波器、過渡電流の抑圧器、及び短絡に対して保護するためのサーキットブレーカを提供する。交流電力ラインの故障に対して保護するために、冗長性のバッテリーパック162と163が、順序正しいシステムシャットダウンを完了しうるような4-1/2分の全システム電力を与える。2個のバッテリーパックの1個162又は163だけが、システムを安全にシャットダウンするために動作するのに必要である。

電力サブシステムは、2つの同一の交流から直流へのバルク電源164と165を備え、これらの電源は、高電力ファクタを備え、1対の36ボルト直流分配バス166と167にエネルギーを供給する。このシステムは、動作中である1個のバルク電源164又は165を用いて、動作し続けることが可能である。

わり、システムは、上述した故障回復モードの1つで動作を続け、故障した電源部品をシステム動作中に交換できる。

この電源システムを、スタンドバイとオフの機能を備えた手動スイッチか、もしくは保守・診断電源の故障の場合に電源オン状態を自動的にオフ状態とする保守・診断プロセッサ170からのソフトウェア制御のもとでのいずれかで、シャットダウンできる。

本発明は、特別な実施例を参照して説明されたが、この説明は、制限的な意味でなされたのではない。開示された実施例の様々な変形が、本発明の他の実施例と同様に、この説明を参照して当業者に明らかである。従って、添付した特許請求の範囲は、本発明の範囲内で実施例の任意のそのような変更を含む。

#### 4. 図面の簡単な説明

第1図は、本発明の一実施例によるコンピュータシステムの電気回路のブロック図、

第2図は、第1図のCPUの電気回路のブロッ

ク図、

第3図は、第2図のCPUに使用されるマイクロプロセッサチップの電気回路のブロック図、

第4図と第5図はそれぞれ、第2図と第3図のCPUにおいて生じるイベントを時間の関数として示すタイミング図、

第6図は、第1図のコンピュータシステムにおける1個のメモリモジュールの電気回路のブロック図、

第7図は、第1図のシステムにおけるメモリバスに対してCPUに生じるイベントを示すタイミング図、

第8図は、第1図のコンピュータシステムでの1個のI/Oプロセッサの電気回路のブロック図、

第9図は、第1図のシステムでのメモリモジュールとI/Oプロセッサの間の転送プロトコルのためのイベントを示すタイミング図、

第10図は、第1図から第3図までのCPUにおける命令の実行のためのイベントを示すタイミング図、

第21図は、第1図、第2図、第3図及び第6図のシステムにおいて使用されるグローバルメモリモジュールのメモリマップにおける専用メモリの位置の説明図、

第22図は、本発明の一実施例によるシステムで使用されるフォールトトレラント電源の回路図である。

11、12、13…プロセッサ(CPU)、

14、15…メモリモジュール、

16…ローカルメモリ、

17…クロック発振器、

21、22、23…バス、

24、25…入出力バス、

26、27…入出力プロセッサ、

28…バス、

29…バスインターフェースモジュール、

30…I/Oコントローラ、

31…ラジアルライン、

32…システムステータスバス、

33…肯定応答/ステータスバス、

第10a図は、第10図の一部の詳細図、

第11図と第12図はそれぞれ、第1図から第3図までのCPUにおける命令の実行のためのイベントを示す第10図と同様なタイミング図、

第13図は、第2図のCPUにおいて用いられる割り込み同期回路の電気回路のブロック図、

第14図、第15図、第16図及び第17図はそれぞれ、第1図から第3図までのCPUでの命令の実行のためのイベントを示す第10図または第11図と同様なタイミング図であり、様々な場面を説明している。

第18図は、第1図、第2図、第3図及び第6図のシステムにおいて使用されるメモリの物理メモリマップ図、

第19図は、第1図、第2図、第3図及び第6図のシステムにおいて使用されるメモリの仮想メモリマップ図、

第20図は、第2図または第3図によるCPUにおけるマイクロプロセッサチップにおける仮想アドレスとTLBエントリのフォーマットの図、

40…マイクロプロセッサチップ、

41、42、43…ローカルバス、

44、45…キャッシュメモリ、

46…浮動小数点コプロセッサ、

50…ライトバッファ、

51…リードバッファ、

52…ライトバッファバイパス、

53…データバス、

54…アドレスバス、

55…制御バス、

56…バスインターフェース、

57…多重アドレス/データバス、

58…コマンドライン、

60…メモリコントローラ、

61…ローカルレジスタ、

62…不揮発性メモリ、

65…割り込み回路、

71…サイクルカウンタ、

72…リフレッシュカウンタ、

73…カウンタ、

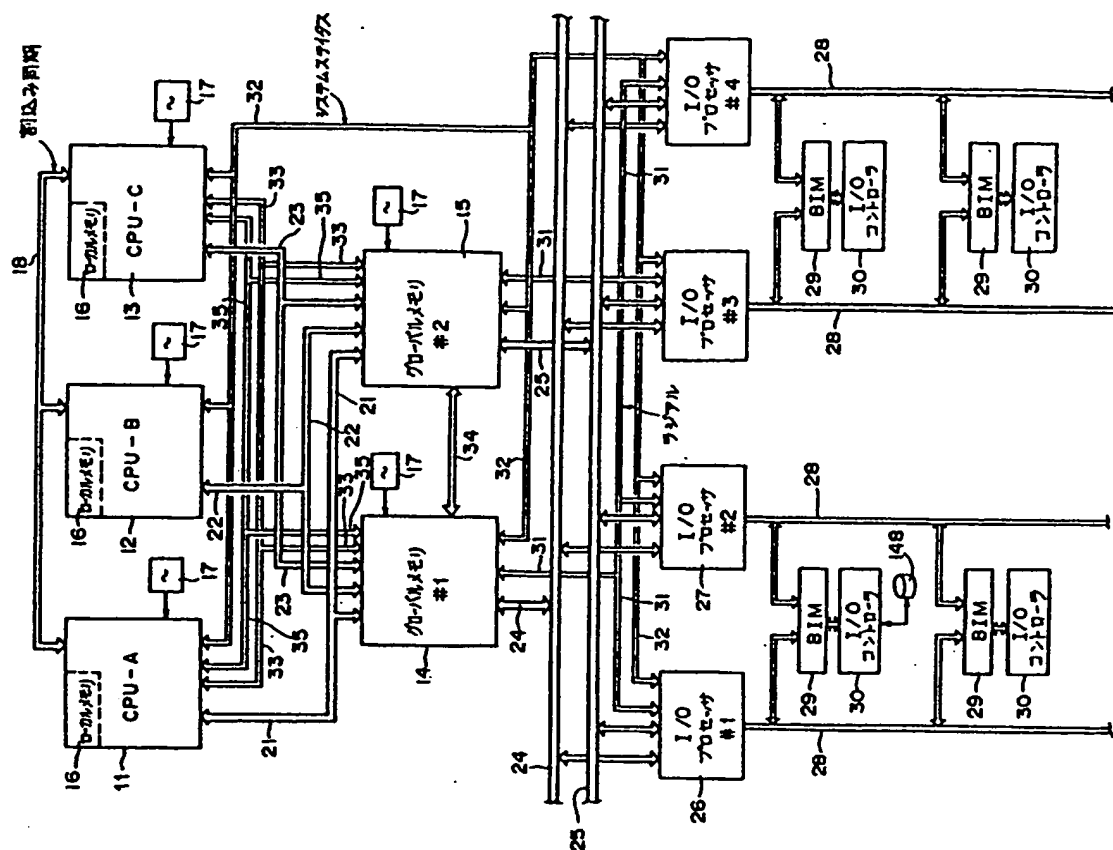
- 74…DMA回路、
- 76…レジスタ、
- 77…ALU、
- 78…シフト、
- 81…プロセッサバス構造、
- 82…命令デコード、
- 83…トランスレーションルックアサイドバッ  
ファ(TLB)、
- 84…仮想アドレスバス、
- 87…パイプライン及びバス制御回路、
- 91, 92, 93…入力/出力ポート、
- 94, 95, 96…レジスタ、
- 100…ポート回路、
- 101…データバス、
- 102…アドレスバス、
- 103…コマンドバス、
- 104…DRAM、
- 105…メモリコントローラ、
- 106…制御・ステークスレジスタ、
- 107…不揮発性RAM、
- 108…ライトプロテクト、
- 109…バスインターフェース、
- 110…アービトラクタ回路、
- 114…リードレジスタ、
- 117…コントローラ、
- 118…リフレッシュカウンタ、
- 119…割り込みポート回路、
- 121, 122…ポート、
- 123…双方向多重アドレス/データバス、
- 124…双方向コマンドバス、
- 126…ステートマシン、
- 127, 128…ラッチ、
- 130…内部ステークス・制御レジスタ、
- 131…バスインターフェース、
- 132…マルチプレクサ、
- 133…制御・データ転送レジスタ、
- 135…割り込み分配器、
- 136…割り込みポート、
- 138…保持レジスタ、
- 141…ローカルメモリエリア、
- 142…グローバルメモリエリア、
- 143…核エリア、
- 144…核データエリア、
- 145…ユーザプログラムページエリア、
- 146…ユーザページエリア、
- 147…ディスクバッファエリア、
- 149…I/Oステージングエリア、
- 160…交流電力ライン、
- 161…電力分配ユニット、
- 162, 163…バッテリーバック、
- 164, 165…バルク電源、
- 166, 167…直流分配バス、
- 168…DC-DCコンバータ、
- 169…システムファン、
- 170…保守・診断プロセッサ、

特許出願人 タンデム・コンピューターズ・

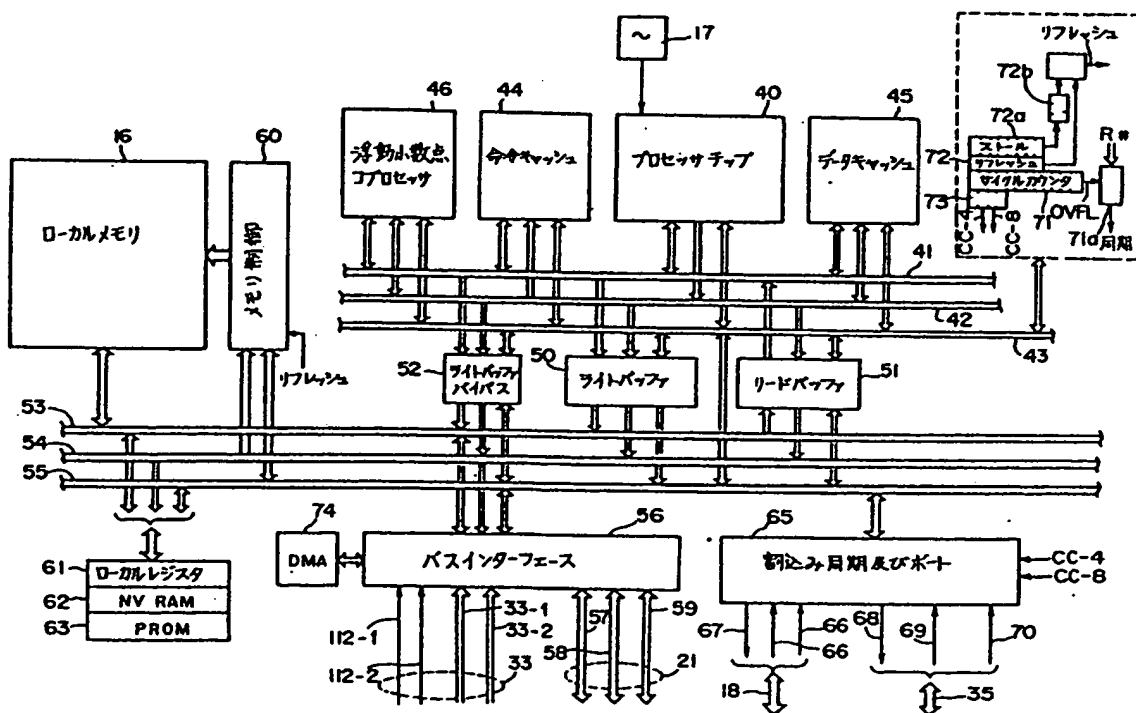
インコーポレイテッド

代理人 弁理士 青山 篠 ほか1名

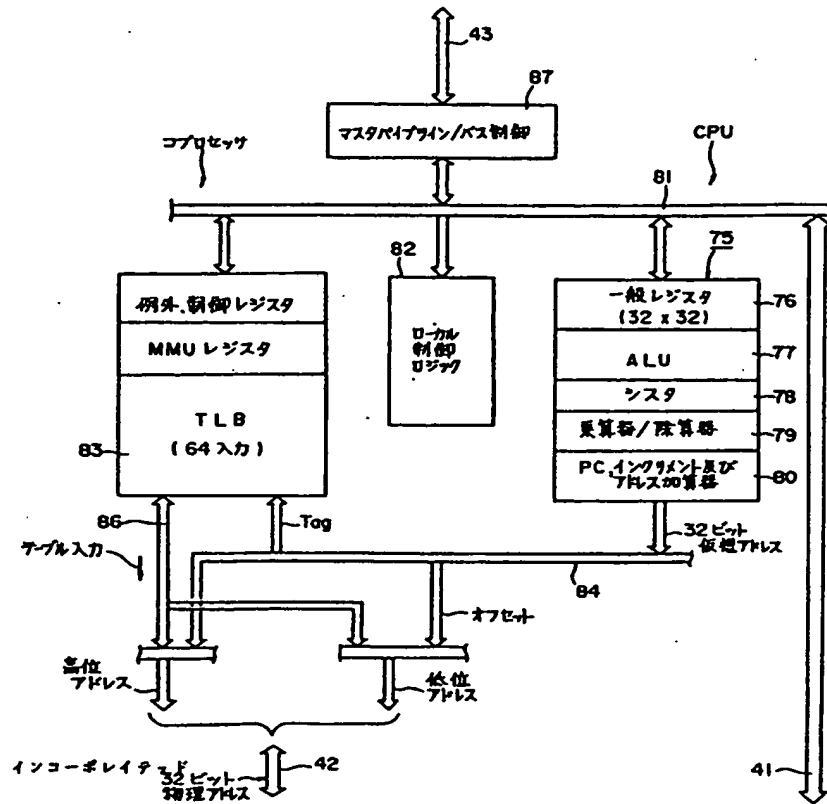
第1図



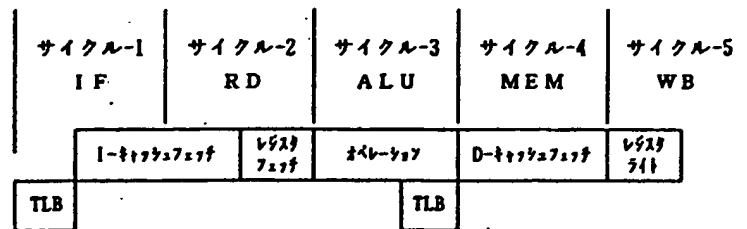
第2図



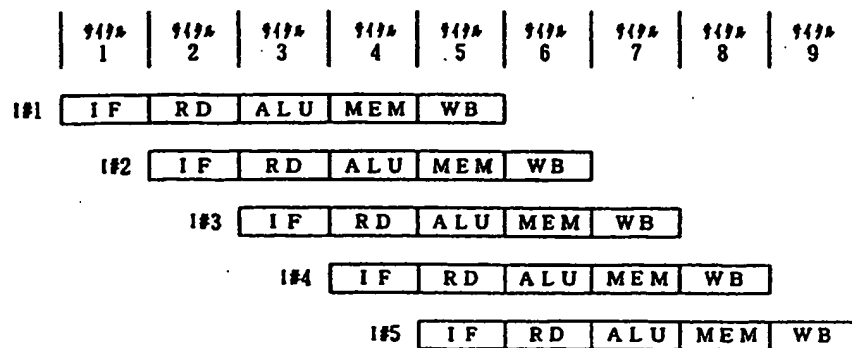
第 3 図

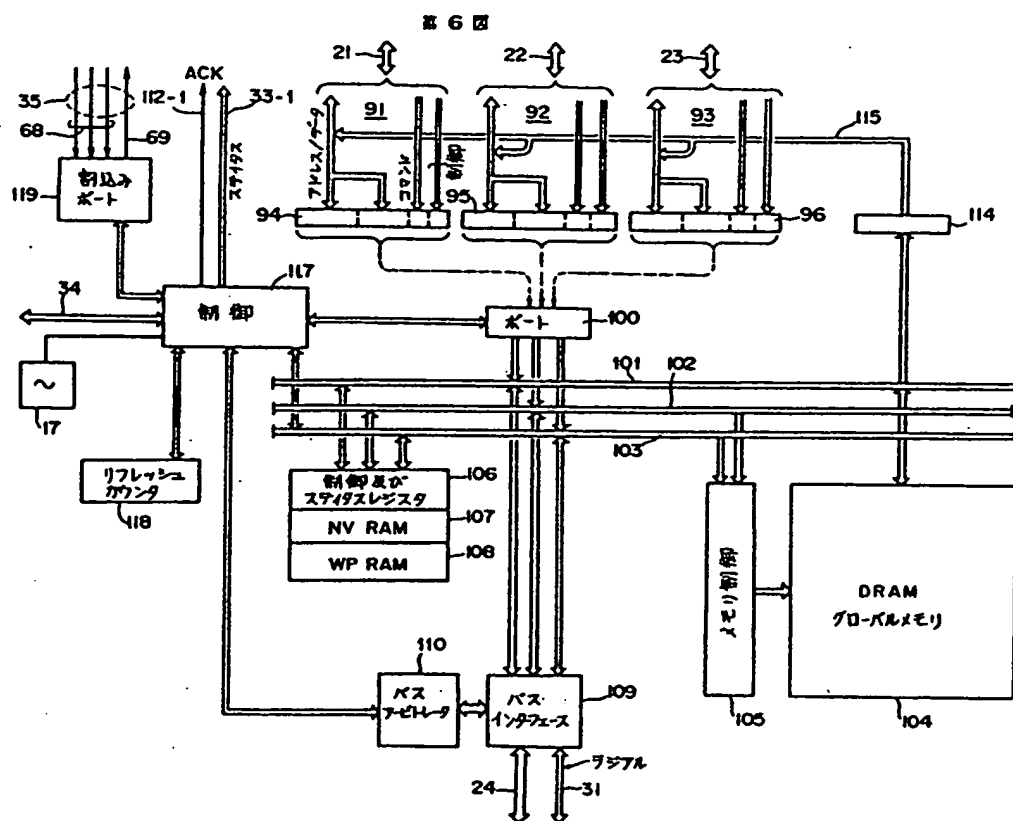


第 4 図

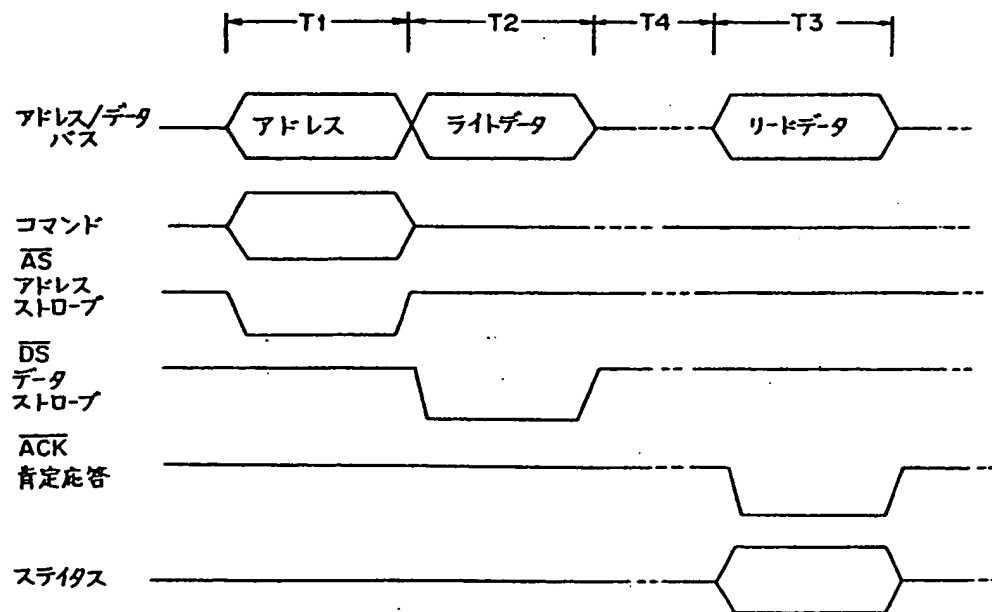


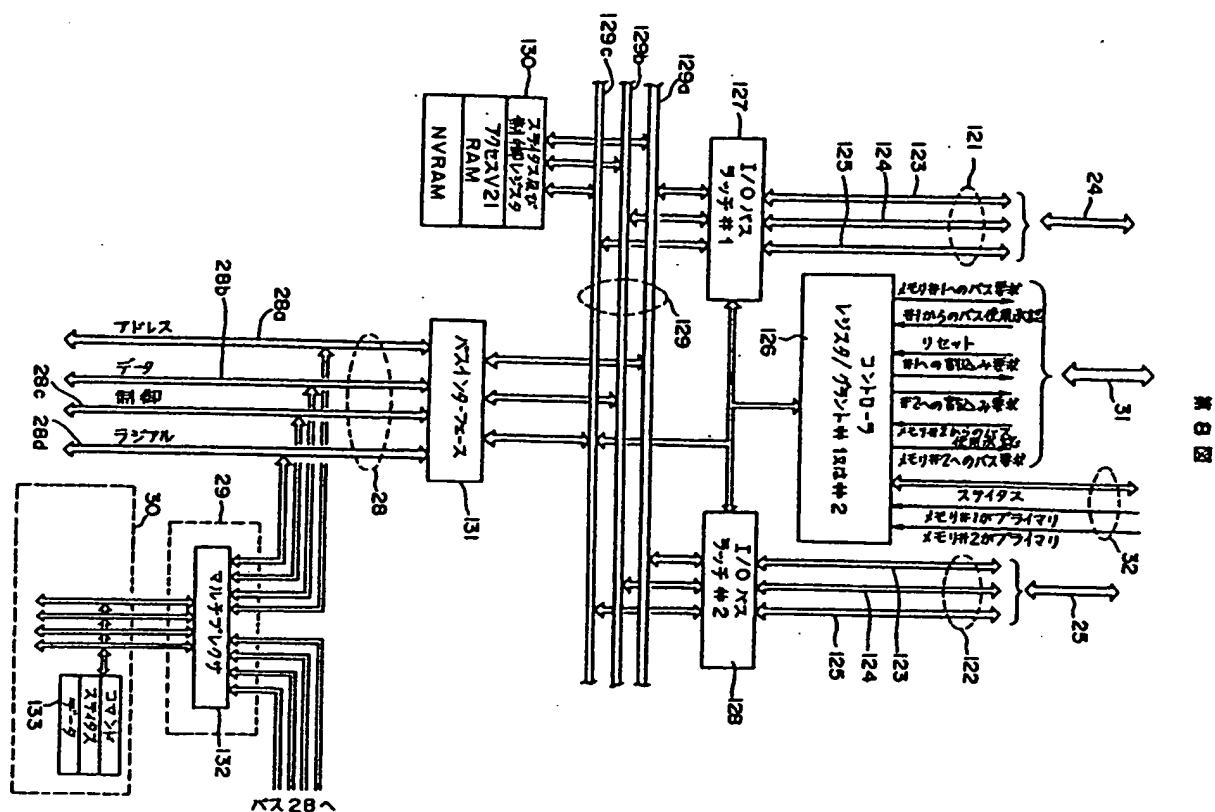
第 5 図





第 7 図





第 9 圖

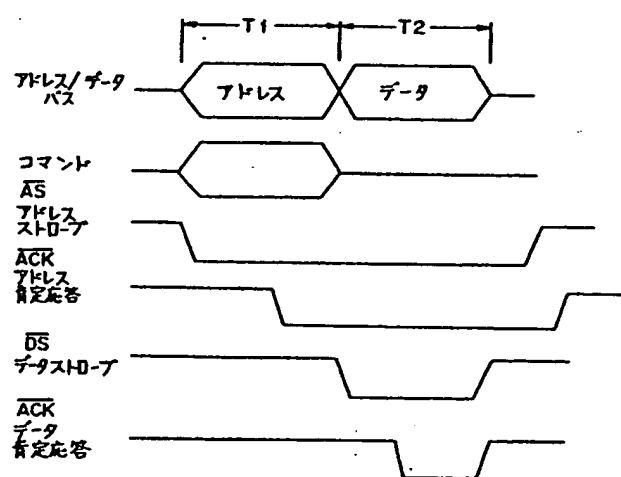


图 9-1 续

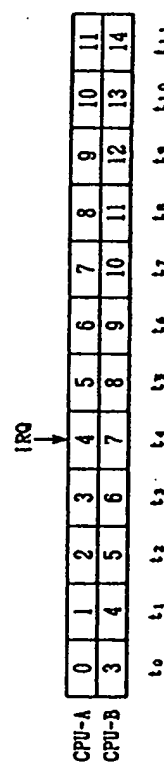
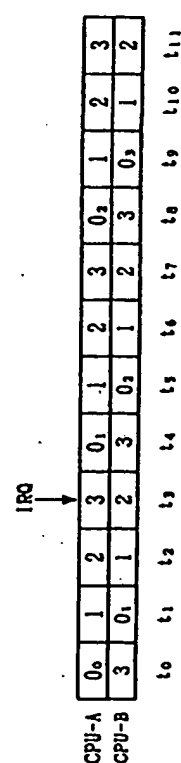


图 2-17 第 17 图



第 10 回

CPU-A	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$
CPU-B	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$	$t_{13}$	$t_{14}$	$t_{15}$	$t_{16}$	$t_{17}$	$t_{18}$
CPU-C	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$	$t_{13}$

第 10a 圖

$l_0$	$l_1$	$l_2$	
	$l_6$	$l_7$	$l_8$
$l_3$	$l_4$	$l_5$	

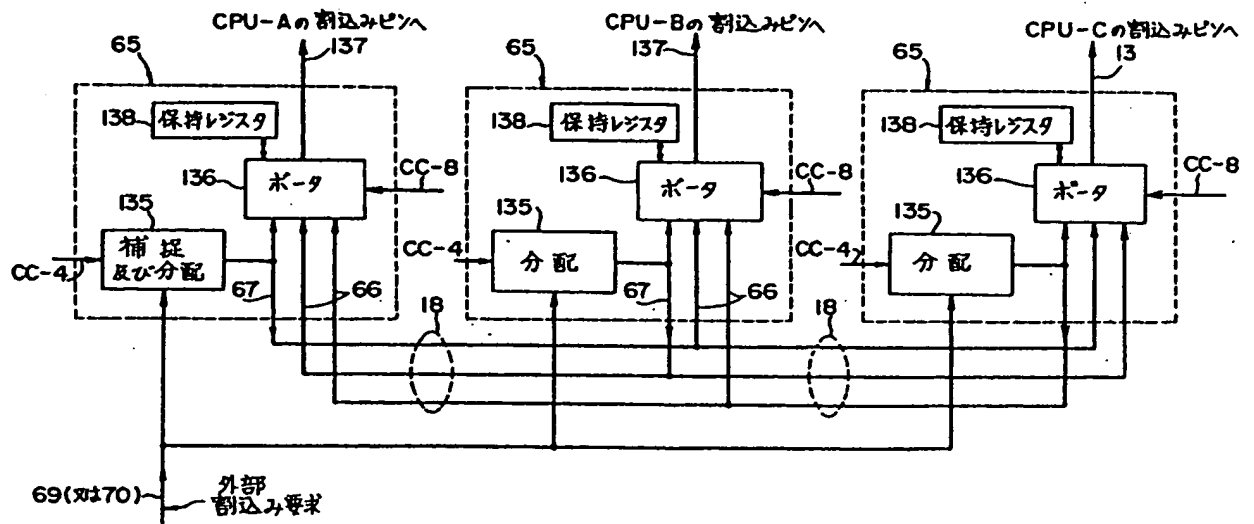
第 11 圖

CPU-A	0	1	2	3	4	4	4	4	4	5	6
CPU-B	2	3	4	4	4	4	4	4	4	5	6
CPU-C	3	4	4	4	4	4	4	4	4	5	6
	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$

第 12 圖

CPU-A	0	1	2	3	4	5	6	7	8	9	10	11
Awb												
CPU-B	3	4	5	6	7	7	7	7	8	9	10	11
Bwb												
	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$

**第13圖**



第 14 図

EX\_IRQ

CPU-A	0	1	2	3	4	5	6	7	8	9	10	11	12
CPU-B	2	3	4	5	6	7	8	9	10	11	12	13	14
CPU-C	3	4	5	6	7	8	9	10	11	12	13	14	15
IRQ_A_PENDING	0	0	0	0	1	1	1	1	1	1	1	1	1
IRQ_B_PENDING	0	0	1	1	1	1	1	1	1	1	1	1	1
IRQ_C_PENDING	0	1	1	1	1	1	1	1	1	1	1	1	1
IRQ_A	0	0	0	0	0	0	0	0	1	1	1	1	1
IRQ_B	0	0	0	0	0	0	1	1	1	1	1	1	1
IRQ_C	0	0	0	0	0	1	1	1	1	1	1	1	1

t<sub>0</sub> t<sub>1</sub> t<sub>2</sub> t<sub>3</sub> t<sub>4</sub> t<sub>5</sub> t<sub>6</sub> t<sub>7</sub> t<sub>8</sub> t<sub>9</sub> t<sub>10</sub> t<sub>11</sub> t<sub>12</sub>

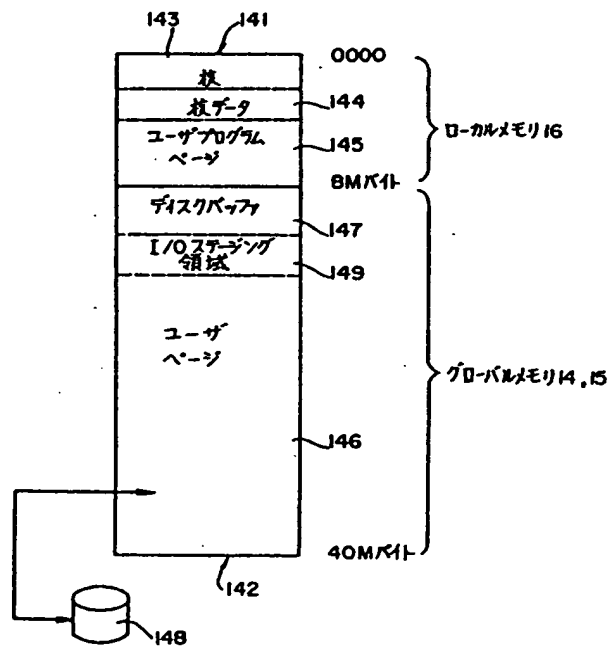
第 15 図

EX\_IRQ

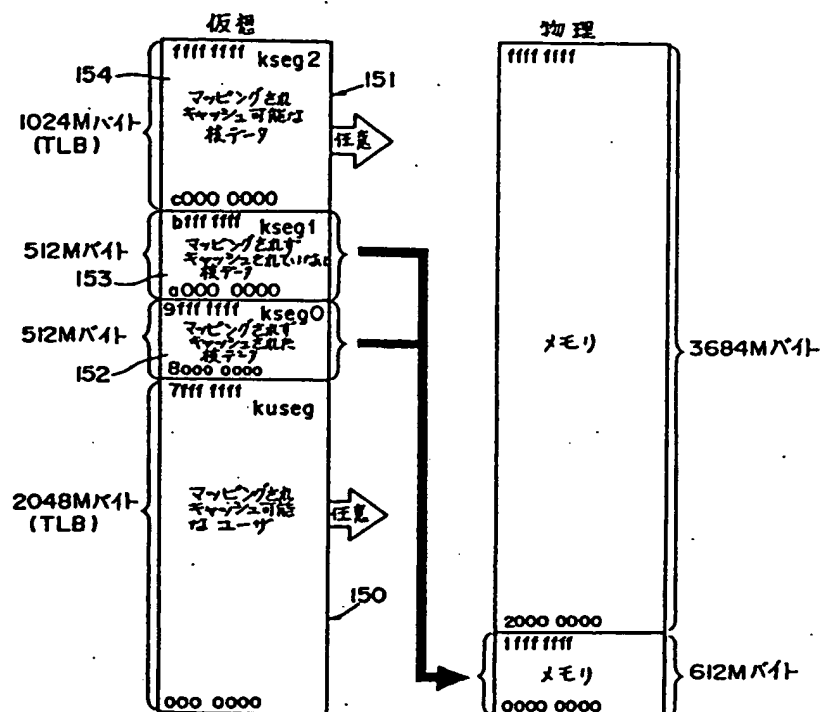
CPU-A	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
CPU-B	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1
CPU-C	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2
EX_IRQ	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
IRQ_A_PENDING	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
IRQ_B_PENDING	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
IRQ_C_PENDING	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
IRQ_A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
IRQ_B	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
IRQ_C	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

t<sub>0</sub> t<sub>1</sub> t<sub>2</sub> t<sub>3</sub> t<sub>4</sub> t<sub>5</sub> t<sub>6</sub> t<sub>7</sub> t<sub>8</sub> t<sub>9</sub> t<sub>10</sub> t<sub>11</sub> t<sub>12</sub> t<sub>13</sub> t<sub>14</sub> t<sub>15</sub>

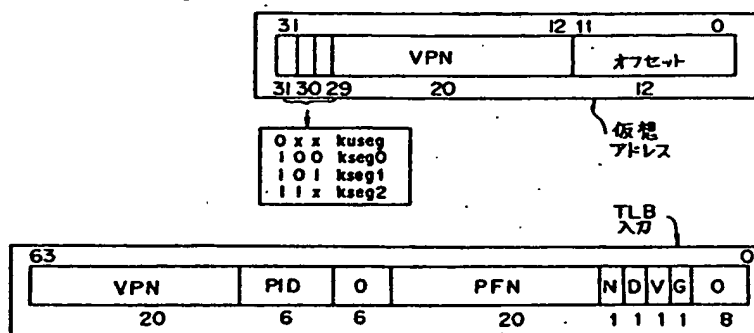
第 18 図



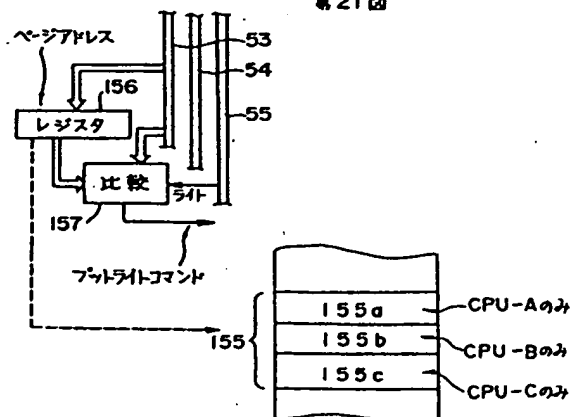
第19図



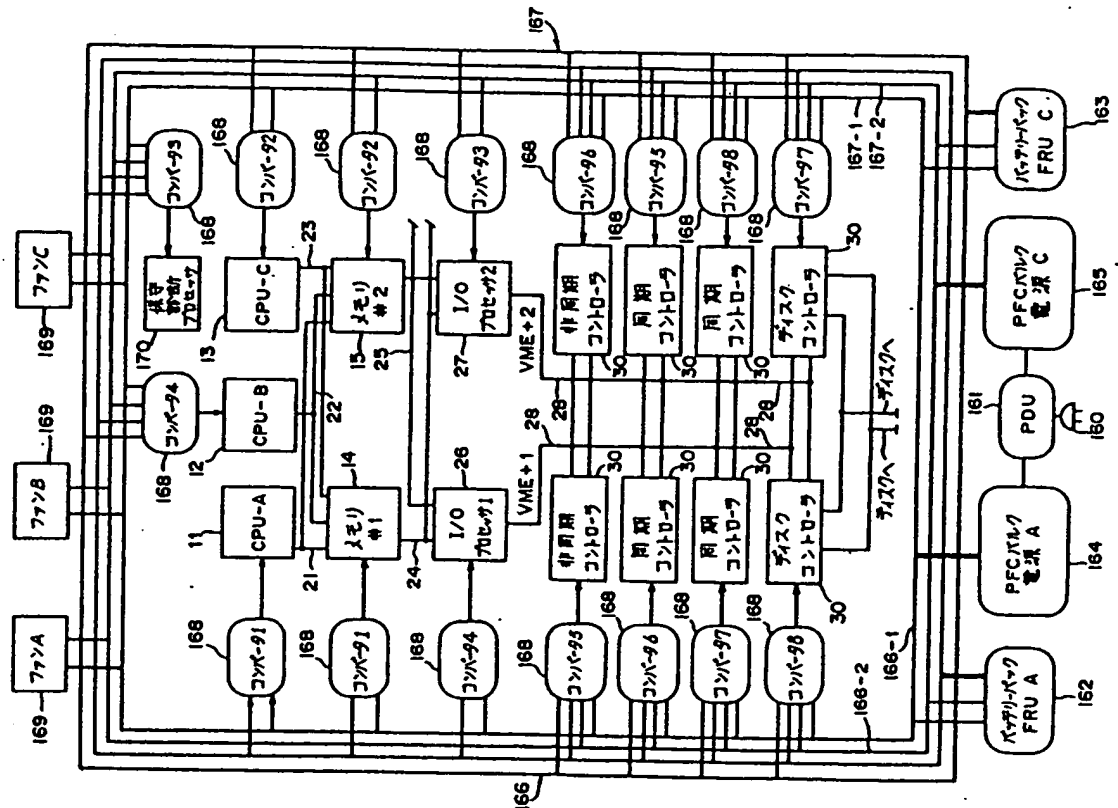
第20図



第21図



第22図



第1頁の続き

優先権主張

◎1988年12月9日◎米国(US)◎282,540

◎1988年12月13日◎米国(US)◎283,573

- |      |                     |               |                                      |
|------|---------------------|---------------|--------------------------------------|
| ◎発明者 | ダグラス・イー・ジュ<br>ウェット  | アメリカ合衆国 78727 | テキサス、オースチン、ウィク<br>リフ・レイン 12401番      |
| ◎発明者 | ケニス・シー・デイベ<br>ツカー   | アメリカ合衆国 78717 | テキサス、オースチン、モノ<br>ナ・コウブ 15702番        |
| ◎発明者 | ニキール・エー・メー<br>タ     | アメリカ合衆国 78758 | テキサス、オースチン、ブライ<br>リエ・ヘン・コウブ 1715番    |
| ◎発明者 | ジョン・デイビッド・<br>アリソン  | アメリカ合衆国 78703 | テキサス、オースチン、ウィン<br>ザー・ロード 1406番、202号  |
| ◎発明者 | ロバート・ダブリュ<br>ー・ホースト | アメリカ合衆国 61821 | イリノイ、シヤンペイン、ロー<br>ブソン・パーク・ドライブ 2804番 |